UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO
FACULTAD DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA, INFORMÁTICA Y MECÁNICA
ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA Y DE SISTEMAS

TESIS

"APLICACIÓN DEL MÉTODO DE AGRUPAMIENTO LINEAL EN INFERENCIA DE REDES GENÉTICAS PROBABILÍSTICAS APLICADO AL PLASMODIUM FALCIPARUM"

PARA OPTAR EL TITULO PROFESIONAL DE :

Ingeniero Informático y de Sistemas

PRESENTADA POR :

Br. Carlos Fernando Montoya Cubas

Br. Juan Christian Yunguri Ttito

ASESOR :

Mgt. Julio Cesar Carbajal Luna

Cusco - Perú

2016

"La vida es demasiado
corta para que nosotros podamos
conseguir de todo, sin embargo debemos
centrar nuestro interés en todo lo que
puede llenar nuestros días"
Bertrand Russell

Resumen

Las reacciones químicas que resultan de la expresión de genes son complejas y aun no se entienden completamente. Se conoce que los genes envían, reciben y procesan la información para formar una compleja red de comunicación, pero la arquitectura y la dinámica de estas redes no es totalmente conocido. De esta forma, un problema importante dentro del campo de la biología sistémica es determinar cómo los genes se relacionan entre si dentro de la célula. Este proceso se conoce como inferencia de redes genéticas.

La inferencia de redes genéticas a partir de datos de expresión es un problema abierto debido a la alta dimensionalidad (número de genes) y al pequeño número de muestras disponibles, incluso considerando el hecho de que las redes sean escasas (numero limitado de genes de entrada por gen objetivo).

De esta forma, podemos encontrar varias técnicas que ayudan a aliviar el problema de la alta dimensionalidad, en este trabajo nos centramos en corroborar la efectividad del método de *Agrupamiento Lineal*, el cual infiere las redes genéticas a partir de particiones en el espacio del reticulado Booleano, inducidos por combinaciones lineales de los valores de los genes predictores. De este modo el número de configuraciones de los valores de los predictores muestran una función lineal en vez de una función exponencial en función al numero de genes, de esta forma es aplicado una selección local de características (genes predictores) para cada gen con el fin de hacer la inferencia.

Este trabajo analiza el método de *Agrupamiento Lineal* aplicado a un conjunto de datos del *microarray* del *Plasmodium falciparum*, uno de los agentes que produce la *malaria*, demostrando la validez de esta técnica en datos reales, habiendo sido capaz de producir conocimiento ya descubierto anteriormente y ayudando a aliviar el problema de dimensionalidad.

Palabras Clave Inferencia de Redes Genéticas . Reducción de la dimensionalidad . Selección de características . Agrupación lineal . Redes Genéticas Probabilísticas

Abstract

The chemical reactions that result in gene expression are complex and not yet fully understood. It is known that genes send, receive and process information to form a complex network of communication, but the architecture and dynamics of these networks are not fully known. Thus, one major problem is to determine how genes are linked within the cell. This process is known as inference of genetic networks.

The inference of genetic networks from gene expression data is an open problem due the large dimensionality (number of genes) and the small number of data samples typically available, even considering the fact that the network is sparse (limited number of input genes per target gene).

In this way, we can find several techniques that help to alleviate the problem of high dimensionality, in this work us focus on confirm the effectiveness of the method of *linear grouping*, it which infer genetic networks starting from partitions in the space of the lattice Boolean, induced by combinations linear of the values of the predictors genes. In this way the number of settings of the values of the predictors shows a linear function instead of an exponential function in function to the number genes, thus it is applied a local selection of features (gene predictors) for each gene in order to make the inference.

this work analyzes the method of *Linear grouping* applied to a set the *microarray data* of *Plasmodium falciparum*, one of the agents that produces the *malaria*, demonstrating the validity of this technique in real data, having been capable of producing knowledge already discovered previously and helping to alleviate the problem of dimensionality.

Keywords Gene Networks Inference · Dimensionality Reduction · Feature Selection · Linear Grouping · Probabilistic Gene Networks

Índice general

Índice de figuras

Índice de cuadros

Lista de símbolos

V	Conjunto de nodos (genes) de una red Booleana
n	Número de nodos (genes) de una red
g_i	i-ésimo gen
i, j	Índices
Φ	Conjunto de funciones Booleanas
ϕ_i	Función Booleana predictora del gen i
ψ	Clasificador
c	Número de clases (rótulos)
v_i	i-ésimo nodo (gen) de una red booleana
t	instante de tiempo
k	Número de genes predictores (grado) de un determinado gen objetivo
k_i	Número de genes predictores (grado) de i-ésimo gen
k^-	Número de genes predictores inhibidores de un determinado gen objetivo
k^+	Número de genes predictores activadores de un determinado gen objetivo
$\langle k \rangle$	Número medio de predictores por gen (grado medio) en una red
v_{ki}	k-ésimo gen predictor que posee arista incidente al gen v_i (objetivo)
$\vec{s}$	Un estado de una red Booleana
$\mathbf{X}$	Vector de características
Y	Variable de clases (rótulos)
$H(\cdot)$	Entropía
$H(Y\|\mathbf{x})$	Entropía condicional de Y dado $\mathbf{X} = \mathbf{x}$
$H(Y\|\mathbf{X})$	Entropía condicional media de Y dado $\mathbf{X}$

α	Parámetro para el criterio de penalización de instancias no observadas de la entropía condicional media
P	Probabilidad
γ	Constante de decaimiento de una ley de potencia
$\mathbf{Z}$	Subconjunto de características de $\mathbf{X}$
m	Número de experimentos temporales
$\mathbf{A}$	Vector de coeficientes de una combinación lineal
a_i	Una de los componentes del vector de coeficientes de una combinación lineal $\mathbf{A}$
$\mathbf{A}^*$	Vector de coeficientes de la combinación lineal que optimice una determinada función criterio
$\mathbf{z}$	Instancia de $\mathbf{Z}$
L	Número entero del mapeo lineal del agrupamiento de configuraciones (instancias) de los predictores
$f(Y = y)$	Frecuencia de observaciones del valor $Y = y$
ECM_{min}	Entropía condicional media mínima
ECM_{AL}	Entropía condicional media por agrupamiento lineal
$\mathcal{F}$	Función criterio

Lista de abreviaturas y términos

BA	Modelo de redes libres de escala (*Scale-Free*) de Barabási-Albert
BEI	Busca exhaustiva incremental
BN	Redes Booleanas (*Boolean Networks*)
DFT	Transformada Discreta de Furier (*Discrete Fourier Transform*)
ECM	Entropía Condicional Media
ER	Modelo de redes aleatorias de Erdös-Rényi
FT	Factor de Transcripción (*Transcription Factor*)
GRN	Redes de Regulación Genética (*Genetic Regulatory Networks*)
IMP	Predicción Intrínsecamente Multivariable (*Intrinsically Multivariate Prediction*
MI	Información Mutua (*Mutual Information*)
PBN	Red Booleana Probabilística (*Probabilistic Boolean Network*)
PGN	Red Genética Probabilística (*Probabilistic Genetic Network*)
RRT	Red de Regulación Transcripcional (*Transcriptional Regulatory Network*)
SAGE	Análisis Serial de Expresión Genética (*Serial Analysis of Gene Expression*)
SBS	Búsqueda Secuencial para Atrás (*Sequential Backward Search*)
SBS-E	Búsqueda Exhaustiva Secuencial para Atrás
SFS	Búsqueda Secuencial para Frente (*Sequential Forward Search*)
SIM	Similaridad
TPC	Tabla de probabilidades condicionales

Capítulo 1

Aspectos generales

1.1. Introducción

Una de las metas de la bioinformática es estudiar y comprender los mecanismos internos del funcionamiento de una célula, como el ciclo celular, diferenciación celular y el metabolismo. El termino bioinformática fue introducido por [Hogeweg, 2011] definido como "el estudio del proceso informático en sistemas bióticos" (del inglés: *the study of informatic process in biotic systems*). Aunque antes de esto fueron realizados algunos trabajos de biología teórica como el trabajo de [Waddington, 1972], también el trabajo de [Kauffman, 1969] con redes booleanas aleatorias para la modelación de la regulación de la transcripción genética. Por otra parte, el área de biología sistemática integra conocimientos de biología para entender como las partes de una célula (genes, proteínas, etc) por ejemplo interactúan entre si con el fin de garantizar el funcionamiento y sobrevimiento de esta célula, es decir en esta área estudiamos el funcionamiento del sistema celular en vez de estudiar sus componentes individualmente. Según [Kitano, 2002] podemos dividir el estudio de la biología sistemática en cuatro niveles:

1. Estructura del sistema: incluye las redes de interacción genética y las redes de vías metabólicas, así como los mecanismos por los cuales tales interacciones modulan las propiedades físicas de las estructuras intra y multicelulares.

2. Dinámica del sistema: como el sistema se comporta en el transcurso del tiempo sobre

diversas condiciones.

3. Control: mecanismo para controlar el estado de una célula para evitar un comportamiento indeseable y mostrar potenciales objetivos terapéuticos para el tratamiento de enfermedades.

4. Proyecto: estrategias para modificar y construir sistemas biológicos con propiedades deseadas.

Dado que una funcionalidad especifica de una célula es fuertemente determinada por los genes que ella expresa, y siendo la transcripción el primer paso en el proceso de transformar la información almacenada en el ADN del organismo en proteínas, es de esperar que este proceso sea regulado por una red de control que coordina la actividad celular [Dougherty, 2011]. Un medio primario de la regulación de la actividad celular es el control de la producción de proteínas a partir de la cantidad de ARN mensajero (ARNm) expresado por los genes. Los mensajes del ARN pasan por los orificios del núcleo celular llegando al ribosoma, en el citoplasma, donde estos son traducidos en secuencias de aminoácidos formando proteínas y construyendo enzimas que catalizan reacciones metabólicas o retornan al núcleo para interactuar con el ADN en la regulación de la síntesis del ARN, [Barrera et al., 2002]. La figura 1.1 ilustra este proceso dentro de la célula. Por lo tanto, conjuntos de genes constituyen redes de comunicación bastante complejas que controlan las vías metabólicas celulares.

De este modo explicar la forma con que esta red se auto regula, es actualmente uno de los principales temas de investigación [Snoep and Westerhoff, 2005], para comprender mejor estos mecanismos regulatorios es considerado la evolución temporal de los niveles de expresión genética, o sea, su dinámica (como el sistema se comporta a medida que avanza el tiempo bajo diferentes condiciones).

En particular, el desarrollo de técnicas masivas de extracción de información molecular, como los ADN *microarrays* [Shalon et al., 1996], SAGE (del inglés: *Serial Analysis of Gene Expression*) [Velculescu et al., 1995] y las RNA-seq (del inglés: *RNA sequencing*)[Wang et al., 2009], están haciendo posible estimar el nivel de expresión de millones

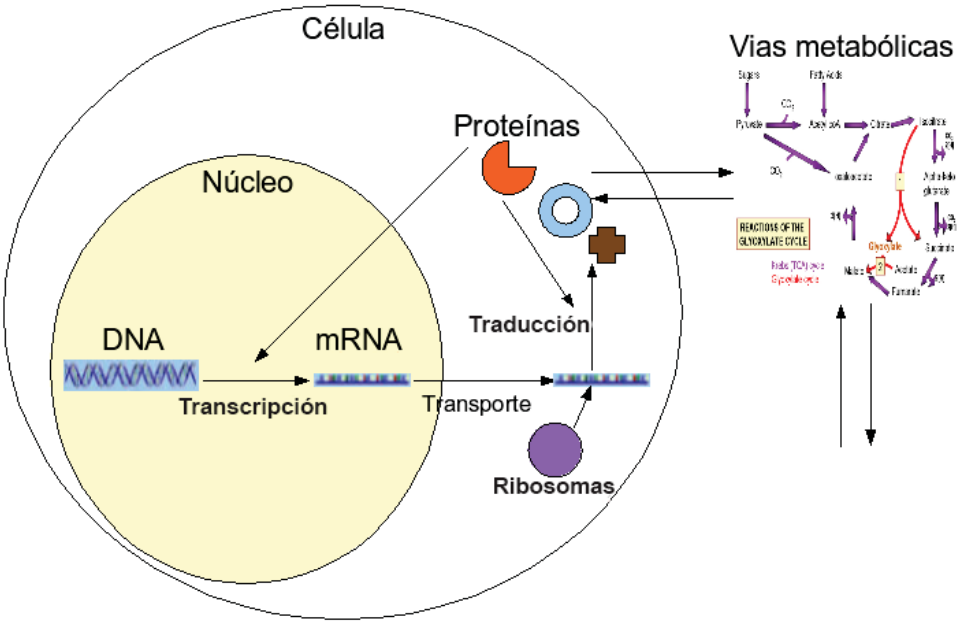


Figura 1.1: Esquema simplificado de la dinámica celular. Fuente [Martins Junior, 2009]

de genes simultáneamente en múltiples instantes de tiempo.

Los genes son elementos importantes en los sistemas de control de organismos, constituyendo una forma de red de comunicación que procesa información biológica y regulan las vías metabólicas de las células, los genes presentan la propiedad de expresarse produciendo copias de ADN en forma de ARN mensajero (del inglés: *messenger RNA* - mARN) [Crick, 1970]. El funcionamiento de un organismo depende de varias vías metabólicas reguladas por redes de expresión genética. En el contexto del análisis de expresión genética, especialmente para el reconocimiento de patrones el conjunto de datos presenta innumerables medidas para pocas observaciones (un número pequeño de muestras experimentales conteniendo millones de genes). Por eso, existe un esfuerzo significativo en el desarrollo de nuevas técnicas computacionales y estadísticas que reduzcan el error de estimación cometido, [Shmulevich et al., 2002, Kelemen et al., 2008, Hecker et al., 2009].

La inferencia de redes a partir de expresiones genéticas es un problema inverso debido al hecho que pueden existir varias redes capaces de generar los datos observados, para ayudar este tipo de problemas a la hora de inferir, están siendo utilizadas diversas metodologías, basadas en diversas áreas de estudio, tales como el reconocimiento de patrones, inteligencia artificial, optimización combinatoria, teoría de información, teoría de control, redes complejas, inferencia estadística, sistemas dinámicos, entre otras.

Una técnica de reconocimiento de patrones comúnmente usada para inferir las redes de regulación genética (del inglés: *Gene Regulatory Networks - GRN*) es la selección de características, en el análisis de señales de expresión genética (concentraciones de mARN), existen dos objetivos principales a la hora de realizar la selección de características según [Dougherty et al., 2008]. Uno de ellos es eliminar los genes irrelevantes de un modelo de clasificación (o predicción) de forma que mejora su desempeño. Otro objetivo es descubrir la estructura de las redes genéticas o de los mecanismos responsables por un determinado fenómeno biológico (por ejemplo, progreso o represión de una enfermedad).

1.2. Identificación del problema

Debido al problema de alta dimensionalidad (un número pequeño de muestras experimentales conteniendo millones de genes) presentada a la hora de inferir datos de señales genéticas están siendo desarrolladas nuevas técnicas computacionales y estadísticas que reduzcan este error de estimación cometido, es por eso que entre los métodos encontrados en la literatura encontramos uno en particular, el método de *Agrupamiento lineal*, fue desarrollado justamente para aliviar el problema de la alta dimensionalidad, este método muestra resultados muy satisfactorios llegando a minimizar tal problema, pero presenta un problema y es que este método como otros a la hora de ser evaluados utilizan datos artificiales para su validación, de esta forma muchas investigaciones se centran en comprobar la eficiencia de estos métodos en datos reales.
De este modo utilizamos datos de expresión genética del transcriptoma del ciclo de desarrollo intraeritrocitario del *"Plasmodium falciparum"* para su respectiva comprobación.

1.3. Objetivos

1.3.1. Objetivo General

Aplicar el método de *Agrupamiento lineal* en inferencia de redes de regulación genética en la vía glicolítica del *"Plasmodium falciparum"* , utilizando el modelo de las redes genéticas probabilísticas y datos de expresión genética del transcriptoma del ciclo de desarrollo intraeritrocitario del *"Plasmodium falciparum"*.

1.3.2. Objetivos Específicos

1. Construir una red genética probabilística (del inglés: *Probabilistic Gene Networks* - PGN), de la vía glicolítica del *Plasmodium falciparum*, utilizando el método de *Agrupamiento lineal*.

2. Evaluar el método de *Agrupamiento lineal*, utilizando datos discretizados de la expresión genética del transcriptoma del ciclo de desarrollo intraeritrocitario del *Plasmodium falciparum*.

1.4. Justificación

El método estudiado de agrupamiento lineal presento resultados eficientes a la hora de minimizar el "problema de la dimensionalidad", estos resultados son mostrados cuando inferimos las redes de regulación genética utilizando datos artificiales [Montoya, 2014], sin embargo falta comprobar la eficiencia de este método en datos reales, de este modo utilizamos datos del *plasmodium falciparum* como instrumento de estudio para comprobar la eficiencia del método de agrupamiento lineal. Tomamos como instrumento de estudio el *plasmodium falciparum* ya que en el Perú se considera una enfermedad re-emergente (alto nivel de emergencia) según afirma la dirección general de epidemiología en su boletín informativo de enero del año 2015 [de Salud, 2015]. El principal problema de los actuales tratamientos de la malaria, es que los medicamentos que antes ya habían conseguido curar esta enfermedad son ahora prácticamente obsoletos, esto por la capacidad de generar resistencia del parásito. Una nueva forma de atacar esta enfermedad es bloquear las proteínas que intervienen en la vía glicolítica del parásito, pues esta vía es la encargada de la producción de energía en la etapa asexual del parásito, si se consigue inhibir estas proteínas el parásito se quedaría sin energía y por consecuencia moriría. Aunque ya existen estudios que tratan de la vía glicolítica del parásito, aún no existen fármacos que actúen para este mecanismo, es por eso entender como actúan los genes relacionados a esta vía es de gran importancia para conseguir su inhibición.

La construcción de la red de regulación genética de la vía glicolítica, proporciona un mapa de posibles genes objetivos donde pueden actuar fármacos que traten esta enfermedad, es

vital conocer cuales de estos genes activan e inhiben los genes que producen las proteínas responsables de generar la energía que necesita el parásito, así también nos ayuda a entender la dinámica de esta red, para poder identificar el momento propicio de inhibir o activar estos genes para bloquear la producción de energía.

El método de agrupamiento lineal ya se mostró eficiente en datos artificiales, la comprobación de esta eficiencia en datos reales utilizando datos del *Plasmodium falciparum*, proporcionaría evidencias biológicas necesarias de este método para poder aplicar en otros tipos de redes de regulación, de este modo se podría examinar otros tipos de enfermedades complejas como son el sida, la esquizofrenia, entre otras, que tienen características genéticas complejas.

1.5. Delimitaciones

Las limitaciones y alcances dentro de las cuales este proyecto se llevará a cabo comprenden:

- No contamos con el apoyo de alguna entidad encargada de proporcionarnos datos discretizados de señales de expresión genética, no existen datos de expresión genética de la vía glicolítica del *Plasmodium falciparum* tomados en Perú, por lo cual para el desarrollo de esta tesis se utilizo un conjunto de datos del *microarray* del *Plasmodium falciparum* tomadas en el país de Honduras proporcionada por los profesores de la UFABC[1] y USP[2].
- Todas las ejecuciones de la aplicación se realizaron en los laboratorios de la IME-USP debido a la gran cantidad de datos que se procesan.

1.6. Metodología

El desarrollo de esta tesis se basa en una metodología de investigación exploratoria con el fin de determinar la naturaleza del problema [Saunders and P, 2007].
El objetivo de esta metodología no es proporcionar respuestas definitivas a las preguntas

[1]Universidad Federal de ABC
[2]Universidad de San Pablo

de investigación, tales como si esta método el cual evaluaremos descubrirá la estructura genética de una determinada enfermedad, por lo tanto en esta tesis solo trataremos de mostrar la efectividad de tal método con datos reales, para lo cual en los experimentos utilizamos datos del ciclo de desarrollo intraeritrocitario del *Plasmodium falciparum*. También adoptamos esta metodología debido a que abordamos nuevos problemas en los que poco o no investigaciones anteriores se han realizado según [RB, 2006].

Por otra lado, debemos tener en cuenta que esta metodología se utiliza como investigación inicial, que constituye la base de una investigación más concluyente, obviamente de llegar a corroborar tal efectividad en datos reales ayudará a seguir haciendo futuras investigaciones.

Capítulo 2

Fundamentación Teórica

En este capitulo presentaremos la fundamentación teórica y revisión bibliográfica necesaria para la lectura del trabajo a ser presentado en el próximo capitulo como este texto se trata de una tesis en computación donde el subárea de actuación es la bioinformática, estaremos presentando tópicos en el área de ciencias exactas y biología:

2.1. Antecedentes

2.1.1. "Selección de Características en Inferencia de Redes de Interacción Genética a partir de conjuntos reducidos de muestras"

La presente tesis contribuyo en adoptar el método de *Agrupamiento lineal* para su posterior evaluación en datos de expresión genética del transcriptoma del ciclo de desarrollo intraeritrocitario del *Plasmodium falciparum*.

Autor: Carlos Fernando Montoya Cubas, tesis presentada para la obtención de Maestría en Ciencias de la Computación de la Universidad Federal del ABC. Año de obtención 2014

Resumen

La inferencia de redes genéticas a partir de datos de expresión es un problema abierto debido a la alta dimensionalidad (número de genes) y el pequeño número de muestras disponibles, incluso considerando el hecho de que las redes son esparcidas. En este trabajo propusieron métodos para minimizar el problema de la dimensionalidad en inferencia de redes booleanas a través de particiones en el espacio del reticulado booleano, inducidas por combinaciones lineales de los valores de los genes predictores. Cada valor de combinación lineal determina una clase de equivalencia entre la configuración de los predictores.

De esta manera, el número de configuraciones de los valores de los predictores se convierte en una función lineal (número de predictores), en lugar de una función exponencial cuando las configuraciones originales son consideradas. Los métodos propuestos siguieron el enfoque de las redes genéticas probabilísticas (del inglés: *Probabilistic Gene Networks* - PGN), que son aplicadas mediante una selección local de características (predictores de genes) para cada gen objetivo al momento de inferir.

Del mismo modo se considera que parte de la información de las configuraciones originales acaba perdiéndose cuando se aplica el agrupamiento, los resultados indican que la inferencia con el agrupamiento lineal tiende a proporcionar redes con mejores similitudes topológicas del que sin el agrupamiento, en casos donde el número de muestras es bastante limitado, de este modo la inferencia realizada trata de tener el mayor número de predictores por gen.

Conclusiones

La propuesta de ese trabajo consistió en el desarrollo de técnicas de selección de características para inferir redes genéticas modeladas por modelos discretos, tales como las redes booleanas, cuyo principio se basa en reducir el número de parámetros de estimación (configuraciones) de los valores de los predictores a través de agrupamientos. Los métodos propuestos consisten en agrupar en una misma clase de equivalencia configuraciones que lleven a un mismo valor de combinación lineal de acuerdo con los coeficientes lineales que optimicen una función criterio adoptada.

Estas agrupaciones también pueden ser vistas geométricamente a través de cortes del reticulado booleano por hiperplanos paralelos, haciendo que dos de estos hiperplanos necesariamente se crucen en un único vértice cada uno, de este modo los vértices poseerán una distancia de Hamming[1] máxima. Estos métodos efectivamente reducen el problema de la dimensionalidad, puesto que el número de clases de equivalencia crece linealmente respecto al tamaño (número de predictores), en lugar de crecer exponencialmente como es el caso del conjunto original de configuraciones. Esto se hace a costa de cierta pérdida de información dado que el conjunto original de configuraciones es mapeado para un conjunto más reducido (proceso semejante a una cuantificación).

El primer método, llamado agrupamiento lineal, siempre aplica el agrupamiento descrito anteriormente, incluso en casos donde todas las configuraciones de predictores son bien observadas. El segundo método llamado agrupamiento selectivo, sólo se aplica a la agrupación si hubiera por lo menos una instancia no observada. Finalmente, el tercer método, denominado agrupamiento mixto, compara el valor de la función criterio obtenida sobre la tabla de probabilidades condicionales sin agrupamiento con el valor de la función criterio obtenida después del agrupamiento lineal, devolviendo así el mejor de ellos.

Los resultados experimentales adoptaron medidas topológicas de similaridad (del inglés: *similarity* - SIM) y el valor de predicción positiva (del inglés: *positive predictive value* - PPV) como medidas de evaluación. Estos resultados demuestran que las inferencias implican un mayor número de predictores por gen, estos tienden a ser beneficiados por el agrupamiento lineal, especialmente si el número de muestras es muy pequeño. Por otro lado, para conjuntos con un mayor número de muestras e inferencias es envolvido un menor número de predictores por gen, en tanto no hay diferencias significativas de desempeño entre agrupar y no agrupar, pues en este caso el número de muestras es suficiente para permitir una estimación estadística razonable a través de las configuraciones originales de los predictores. Esto sugiere que los métodos propuestos pueden ser mejorados mediante un análisis de multiresolución en el que los únicos predictores que tienen un escaso número de observaciones sean agrupados. El análisis multiresolución se ha empleado con éxito, por ejemplo, en el diseño de filtros binarios [Dougherty et al., 2001].

[1] número de diferencias entre dos secuencias binarias

Una desventaja de este enfoque del agrupamiento lineal es que ella crea clases de equivalencias desequilibradas. Por ejemplo, dado un reticulado booleano de 3 dimensiones (3 características), dos clases tendrán apenas una configuración cada una, mientras que las otras dos clases tendrán tres configuraciones cada una. Y este desequilibrio se agrava con el aumento de dimensión. Para reducir este problema, se puede considerar una amplia gama de posibles valores para los coeficientes lineales, ya que en este trabajo, fueron considerados apenas dos valores posibles para los coeficientes lineales: -1 (inhibición) y $+1$ (activación). Aumentar el número de posibles valores de estos coeficientes significa asignar una variable de peso para la activación o inhibición de un predictor. Por ejemplo, un predictor con coeficiente $+2$ tendría el doble de la influencia activadora de un predictor con coeficiente $+1$ así como un predictor con coeficiente -2 tendría el doble de la influencia inhibitoria de un predictor con coeficientes -1. La desventaja de tener en cuenta un mayor número de estos valores es que el número de clases de equivalencia puede aumentar hasta tal punto que el poder de estimación de la tabla de frecuencias prácticamente no aumentaría en relación a la tabla de las configuraciones originales. Además, un número mayor de valores para los coeficientes lineales implicaría un aumento en la complejidad computacional, puesto que un mayor número de combinaciones lineales deberán ser evaluados.

Para más información de esta tesis [Montoya, 2014].

2.1.2. "Selección de Características y Predicción Intrínsicamente Multivariada en Identificación de Redes de Regulación Genética"

La presente tesis contribuyó en adoptar la función criterio, la función criterio adoptada fue la "entropía condicional media con penalización de instancias no observadas" para su posterior evaluación del método de *Agrupamiento lineal* aplicado a datos de expresión genética del transcriptoma del ciclo de desarrollo intraeritrocitario del *Plasmodium falciparum*.

Autor: David Correa Martins Junior, tesis presentada para la obtención de Doc-

tor en Ciencias de la Computación de la Universidad de São Paulo. Año de obtención 2008

Resumen

Este trabajo analiza aspectos de la selección de características en el problema de identificación de redes de regulación genética a partir de señales de expresión genética. Particularmente, propone un modelo de redes genéticas probabilísticas (del inglés: *Probabilistic Genetic Network* - PGN) que devuelve una red construida a partir de la aplicación recurrente de algoritmos de selección de características orientados por una función criterio basada en entropía condicional. Tal criterio incluye la estimación del error por penalización de muestras raramente observadas. Los resultados de este modelo se aplican a datos sintéticos y al conjunto de datos de *microarray* del *Plasmodium falciparum*, uno de los agentes que causa la malaria, se demuestra la validez de esta técnica habiendo sido capaz de no solamente reproducir conocimientos ya producidos anteriormente, sino también el de producir nuevos resultados.

Otro aspecto investigado en esta tesis es el fenómeno de la predicción intrinsicamente multivariada (del inglés: *Intrinsically Multivariate Prediction* - IMP), o sea, el hecho de que un conjunto de características sea un óptimo caracterizador de los objetos en cuestión, pero cualquiera de sus subconjuntos propiamente contenidos no consiguen representarlos de forma satisfactoria. En este trabajo las condiciones para el surgimiento de este fenómeno fueron obtenidas de forma analítica para conjuntos de 2 y 3 características en relación al gen objetivo.

En el contexto de redes de regulación genética, fueron obtenidas evidencias en que los genes objetivos del conjunto IMP poseen un enorme potencial para ejercer funciones vitales en sistemas biológicos. El fenómeno conocido como canalización es particularmente importante en este contexto. En datos de *microarray* del melanoma, constatan que el gen DUSP1, conocido por ejercer la función canalizadora, fue aquel que obtuvo el mayor número de conjuntos de genes IMP, haciendo que todos ellos posean lógicas de predicción canalizadoras.

Además de eso, las simulaciones computacionales para la construcción de redes con 3 o más genes muestran que el tamaño del territorio de un gen objetivo puede tener un im-

pacto positivo en su contenido de IMP con relación a sus predictores. Esta puede ser una evidencia que confirma la hipótesis de que los genes objetivos del conjunto IMP poseen la tendencia de controlar diversas vías metabólicas cruciales para el mantenimiento de las funciones vitales de un organismo.

Conclusión

Este trabajo presento resultados en dos frentes de la investigación que involucra la selección de características para tratar problemas de modelado de redes de regulación genética. Uno de esos frentes lidio con el problema de la identificación de redes genéticas, visto como uno de las etapas más importantes en el largo y complejo proceso del análisis de datos genómicos. En este contexto, se propuso un método de identificación de redes genéticas probabilísticas asumiendo datos de expresión genética temporal obtenidos de los *microarrays* como un proceso que sigue el primer orden de la cadena de Markov. Debido a la falta de un número satisfactorio de muestras, este modelo considera que existe solamente una función de transición de estado para todos los momentos de tiempo considerados (por la invarianza de desplazamiento), que permite la aplicación de algoritmos de selección de características para evaluar la dependencia entre los genes. Para determinar dicha dependencia, se propuso un criterio basado en la función de la entropía condicional que penaliza a las instancias que se observan solamente una vez en el conjunto de datos, incorporando la estimación del error directamente en los criterios. Los resultados experimentales obtenidos a través de simulaciones y análisis de datos del *microarray* del *Plasmodium falciparum* utilizo genes objetivos de dos módulos vitales del parásito, los cuales demuestra la validez de esta técnica con el objetivo de inferir nuevos conocimientos biológicos.

Otro principal objetivo de esta tesis fue sobre la predicción intrínsecamente multivariada, un fenómeno bastante importante que ocurre cuando se analiza conjuntos de características para la predicción del comportamiento de una variable objetivo (clase). Tal fenómeno es uno de los principales responsables por el hecho de no existir, hasta el presente, un algoritmo de selección de características que garantice la optimización sin evaluar exhaustivamente todos los conjuntos de características. Además de esto, en el contexto de redes de regulación genética, los genes IMP pueden ser de vital importancia como controladores de diversas vías metabólicas.

Para más información de esta tesis [Martins Junior, 2009].

2.1.3. "Construcción de las redes genéticas probabilísticas del *Plasmodium falciparum* en las señales de expresión dinámica del ciclo de desarrollo intraeritrocitario"

El presente articulo contribuyo en adoptar el modelo a utilizar en las redes de regulación genética, el modelo adoptado fue el de las "Redes Genéticas Probabilísticas" para su posterior evaluación del método de *Agrupamiento lineal* aplicado a datos de expresión genética del transcriptoma del ciclo de desarrollo intraeritrocitario del *Plasmodium falciparum*.

Autores: Junior Barrera, Roberto M. Cesar Jr., David C. Martins Jr., Ricardo Z.N. Vêncio, Emilio F. Merino, Márcio M. Yamamoto, Florencia G. Leonardi, Carlos A. de B. Pereira and Hernando A. del Portillo, articulo presentado por los investigadores del IME-USP y BIOINFO-USP en el libro *"Methods of Microarrays Data Analysis V"*. Año de publicación 2007

Resumen

La terminación de la secuencia del genoma del *Plasmodium falciparum* reveló que cerca del 60 % del genoma corresponden a las proteínas hipotéticas y que muchos genes, en las cuales cuyas rutas metabólicas o productos biológicos son conocidos, no son predecidos a partir de búsquedas de similitud de secuencia. Recientemente, utilizando la expresión genética global de la fase asexual del *Plasmodium falciparum* en 1 h en la escala de resolución y la transformación discreta basada en técnicas de Fourier, se ha demostrado que muchos genes están regulados de manera periódica sola durante la fase asexual. Por otra parte, ordenando a los genes según la fase de expresión, se generó una nueva lista de objetivos para el desarrollo de vacunas y medicamentos. En el articulo, se describen los genes bajo una perspectiva diferente: una lista de propiedades funcionales se atribuye a las redes de genes que representan subsistemas del sistema de regulación de la expresión del *Plasmodium falciparum*. El modelo desarrollado para representar redes genéticas, llamada

Red Genética Probabilística (del inglés: *Probabilistic Genetic Network* - PGN), la cual es una cadena de Markov con algunas propiedades adicionales. Este modelo imita las propiedades de un gen como una puerta estocástica no lineal y son construidos los sistemas de acoplamiento de estas puertas. Por otra parte, la minería de expresión dinámica de señales para las técnicas de diseño de la PGN, muestra diferentes bases de datos y conocimientos biológicos, en las cuales se desarrolló una herramienta que las integra. La aplicabilidad de esta herramienta fue descubrir redes genéticas del sistema de regulación de la expresión de la malaria a la vez ha sido validada mediante el camino glicolítico como un "estándar de oro", así también la creación de una PGN del apicoplasto.

Conclusiones

Con el fin de avanzar en el conocimiento sobre la biología del *Plasmodium falciparum*, fue diseñado las PGNs de las señales de expresión dinámica de la fase asexual reportado por [Bozdech and DeRisi, 2005]. A diferencia de su enfoque en la transformada discreta de Furier (del inglés: *Discrete Fourier Transform* - DFT), el diseño de la PGN permitió que se utilice todos los elementos disponibles en la base de datos. Significativamente, esta técnica se aplicó a los genes objetivos que codifican enzimas del camino glicolítico y se obtuvo una red glicolítica biológicamente significativa. Luego, se aplico este algoritmo para construir una PGN del apicoplasto y aunque los genes "semillas" del apicoplasto fueron encontrados, muchos otros genes carecen de la secuencia de señal bipartito péptido. Estos resultados se obtuvieron sin tener en cuenta la equivalencia entre las combinaciones lineales de los insumos, lo que debería mejorar los resultados, ya que reducirá los errores de estimación y la hipótesis es bastante coherente con la dinámica observada del gen. Además, este modelo permitió distinguir entre las señales inhibitorias y excitatorias. Aunque la transformación normal crea las clases de equivalencia que disminuye los errores de estimación, estos amplifican el ruido en los genes de limpieza que tienen señales de expresión casi constante. Una forma de eludir este problema es detectar y excluir los genes de la limpieza del sistema regulatorio antes de la cuantización de la señal. Los próximos pasos de la investigación incluyen principalmente la mejora de la técnica de diseño de red y la validación a través de la genética reversa acerca de algunos genes no predecidos previamente por otros algoritmos como parte del apicoplasto. Si se valida, se podría utilizarse

el enfoque PGN para anotar los genes no considerados por el enfoque DFT y acelerar el descubrimiento de nuevas curas contra la malaria.
Para más información de este articulo [Barrera et al., 2007].

2.2. Fundamentos de la genética y biología molecular

La herencia biológica que un individuo transmite a su descendencia se denomina genotipo y está codificada por una secuencia de nucléotidos, en una macromolécula de ácido desoxirribonucleico (ADN). El ADN almacena dicha información en una serie de sub unidades elementales de información, denominadas genes. El conjunto de todos los genes que constituyen el genotipo de un ser vivo concreto se denomina genoma. El estudio del origen, funcionamiento y evolución del genoma, es mediante diferentes técnicas y disciplinas que se conoce como genómica [Datta and Dougherty, 2009]. El genotipo almacenado en el ADN contiene instrucciones para la síntesis de proteínas, moléculas funcionales encargas de controlar el metabolismo celular. Sin embargo, la información almacenada en el ADN suele permanecer intacta en el núcleo de las células. La síntesis proteica precisa de una molécula intermedia a la cual se transfiere la información contenida en el ADN. Esta molécula en la que se transcribe la información genética es el ácido ribonucleico (ARN), que posteriormente se traduce en una proteína. Cuando la información almacenada en un gen es finalmente transcrita a una molécula de ARN, se dice que el gen se ha expresado. El fenotipo es el resultado de la expresión del genotipo. Este mecanismo de almacenaje, transcripción y traducción de la información genética se conoce como dogma central de la biología molecular. Los avances tecnológicos de los últimos veinte años han permitido desarrollar métodos y técnicas experimentales capaces de cuantificar la expresión genética.

2.3. Conceptos de biología molecular

Los monómeros son moléculas de pequeña masa molecular, simples y estables que constituyen unidades básicas de construcción en Biología Molecular, por ejemplo: monosacáridos, ácidos grasos, nucleótidos o aminoácidos. Los monómeros suelen aparecer unidos entre si, mediante enlaces covalentes, formando grandes estructuras llamadas polímeros. Si el

polímero está formado por un único tipo de monómero, se denomina homopolimero. Si la estructura posee diferentes tipos de moléculas, se llama heteropolímero o copolímero. Los nucleótidos son polímeros formados por la unión de una base nitrogenada, un monosacárido y ácido fosfórico. La unión entre la base nitrogenada y la pentosa se hace a través del grupo hidroxilo del carbono 1', formando una estructura que se conoce como nucleósido. Este nucleósido se une a un grupo de fosfato a través del carbono 5' de la pentosa y se conoce cómo nucleótido-monofosfato (NMP), si existen dos grupos de fosfatos se denomina nucleótido-difosfato (NDP) y para tres grupos de fosfatos se denomina nucleótido-trifosfato (NTP). El ácido nucleico es un copolímero formado por diferentes nucleótidos unidos entre si mediante enlaces fosfodiéster (enlace entre el carbono 5' de la base sacárida a un ion fosfato y éste, a su vez, al grupo hidroxilo del carbono 3' de otro monosacárido)[Castillo, 2012].

2.3.1. Ácidos nucleicos

Los ácidos nucleicos son cadenas polinucleótidas de diferente longitud, que adoptan diferentes estructuras. Existen dos tipos de ácidos nucleicos: ácido desoxirribonucleico (ADN) y ácido ribonucleico (ARN). Aparte de la composición sacárida y otras diferencias estructurales, la característica principal que distingue a ambos ácidos nucleicos es el conjunto de bases nitrogenadas que los componen.

ADN

El ADN está compuesto por dos cadenas de nucleótidos, constituidas por largas secuencias de cuatro bases nitrogendas: Adenina (A), Guanina (G), Citosina (C) y Tiamina (T). Las dos hebras están unidas entre si por puentes de hidrógeno entre los nucleótidos, con una peculiaridad: la Adenina se empareja con la Tiamina a través de dos enlaces, mientras que la Guanina lo hace con la Citosina mediante tres enlaces. Los pares A=T y G=C poseen el mismo tamaño y la molécula de ADN se enrolla adquiriendo la estructura de doble hélice que la caracteriza. Sin embargo, el ADN se organiza en una serie de estructuras de mayor nivel que compactan dicha molécula en el interior de la célula. La unidad estructural inmediatamente superior a la cadena de doble hélice es la Cromatina, compuesta por una serie de complejos proteicos, entorno en los que se enrolla la cadena bicatenaria del ADN.

La cromatina se organiza en estructuras más complejas hasta compactar la molécula de ADN en una estructura superior denominada cromosoma. La secuencia de nucleótidos a lo largo de la cadena del ADN codifica la información sobre una proteína. Sin embargo, no es la responsable de su síntesis, simplemente almacena la información necesaria sobre la cantidad y el momento de su producción. Esta parte del código que regula por completo la síntesis de una proteína se denomina gen y es la unidad de almacenamiento de la información hereditaria. La información total codificada por toda la secuencia del ADN se denomina genoma.

ARN

El ARN está compuesto por una cadena monocatenária de cuatro nucleótidos: A, G, C y Uracilo (U). El ARN, se diferencia del ADN no sólo en su composición química (el ARN se basa en ribosa en lugar de desoxirribosa y la base complementaria a la Adenina es el Uracilo, en lugar de la Tiamina) sino también en su estructura. El ARN aparece como una cadena simple que puede plegarse sobre si misma adoptando diversas formas. En comparación con el ADN, las secuencias de nucleótidos que lo componen son mucho más cortas. Sin embargo, la característica principal de esta molécula es su gran actividad en el desarrollo celular que, tras diferentes procesos de maduración, es la responsable final de la síntesis proteica.

2.3.2. Dogma central de la biología molecular

El ADN almacena la información necesaria para la producción de una proteína, pero no participa directamente en su síntesis. Para ello, la información almacenada en el ADN se transcribe a otro tipo de ácido: ARN mensajero (ARNm) que, posteriormente, se encargará de sintetizar la proteína en otro proceso denominado traducción. Cuando un gen se ha transcrito a ARN, se dice que se ha expresado. Este mecanismo de codificación, transcripción y traducción de la información genética se conoce como el dogma central de la biología molecular. 2.1

Figura 2.1: Diagrama de flujo de la información genética que representa el mecanismo de codificación y transmisión de la herencia biológica. El dogma central de la Biología Molecular establece un mecanismo unidireccional, en el cual, la información hereditaria codificada en el ADN se replica y se transmite a la descendencia. La transcripción del ADN a ARN es la primera fase del proceso de expresión genética que finaliza con la traducción del transcrito a una proteína funcional. Por otro lado, existen procesos especiales que introducen relaciones inversas como la síntesis de ADN complementario a partir de ARN, figura adoptada de [Castillo, 2012].

Transcripción del ADN: ARNm

El proceso de transcripción genética es muy complejo y pueden distinguirse hasta cinco etapas diferentes:

- Preiniciación

  El primer paso consiste en localizar la región en la que debe comenzar la transcripción. Por ello, a cada gen le precede una secuencia de ADN no codificante (del inglés: *untranslated region* - UTR) conocida como región promotora y que identifica el inicio del gen. Para marcar este punto, existe un conjunto de proteínas funcionales llamadas factores de transcripción (del inglés: *Trancription Factor* - FT), que localizan y reconocen estas secuencias UTR y se adhieren al carbono 5' inmediatamente anterior al gen, formando un complejo de pre iniciación que establece el punto de inicio de la transcripción genética. Las secuencias UTR constituyen elementos reguladores fundamentales en una primera etapa del proceso de transcripción. La longitud de estas regiones depende del organismo y del gen considerado, oscilando entre las 500 bases (para organismos más simples, como la levadura) hasta 2000 bases (para

especies más complejas, como el ser humano).

- Iniciación

  Una vez localizado el promotor, el complejo de pre iniciación se cierra con la unión de la ARN polimerasa (ARNp): una enzima que cataliza la polimerización de ribonucleótidos, uniéndolos mediante enlaces fosfodiéster. En este momento, uno de los FT (la helicasa, una enzima capaz de romper puentes de hidrógeno) desnaturaliza parcialmente el ADN, separando la doble hélice a lo largo de dieciocho pares de bases nitrogenadas, formando un complejo abierto conocido como burbuja de transcripción.

- Disgregación del promotor

  Una vez sintetizado el primer enlace fosfodiéster, el complejo de preiniciación se desprende del promotor. La separación se debe a otro FT, la quinasa, una enzima que fosforilíza la ARNp.

- Elongación

  Los ribonucleótidos se aparean complementariamente a la secuencia de ADN, mediante enlaces por puente de hidrógeno. Una vez apareados, el centro activo de la ARNp sintetiza el enlace fosfodiéster entre dos ribonucleótidos, formando la cadena de ARN complementaria, proceso conocido como elongación

- Terminación

  El proceso de elongación continúa hasta que una determinada secuencia de ARN sea sintetizada. Esta secuencia esta situada en el extremo del gen y es rica en Adenina y Citosina seguida de Tiamina. Una vez sintetizada, esta estructura desestabiliza el complejo ADN-ARN, separándolas, liberando la ARNp y renaturalizando la molécula de ADN.

2.3.3. Postprocesamiento del ARNm: maduración

El proceso de transcripción de ADN a ARNm se produce en el núcleo celular y es exclusivo de organismos eucariotas. Hay otra característica importante que diferencia el ARNm procariota del eucariota: el ARNm eucariota contiene secuencias que no codifican información

(secuencias no codificantes) llamadas intrones, que separan las secuencias que si contienen información genética, llamadas exones. El ARNm procariota solo contiene secuencias codificantes. Por ello, el ARNm eucariota sufre un postprocesado del tránscrito primario, que se denomina maduración del ARNm. Antes de ser transportado al citoplasma (en ocasiones, incluso antes de haber transcrito por completo la cadena de ARNm) se añade al extremo 5' de la secuencia un nucleótido modificado: la 7-metil-guanina (metilG). Esta molécula, denominada caperuza (CAP) aporta estabilidad al tránscrito. Una vez finalizada la transcripción, se añade al extremo 3' una cola de poliadenilato (poli-A, secuencia de ARN basada en Adenina) que protege el ARNm de la degradación y ayudará a su transporte. Existe una secuencia de poliadenilación (AAUAAA) que se sintetiza unos veinte nucleótidos antes de la secuencia de terminación. Tras el encapuchado y poliadenilizado del transcrito primario, se procede a la eliminación de secuencias no codificantes, proceso conocido como empalme o splicing. El proceso se lleva a cabo por un complejo (conocido como spliceosoma) formado por pequeñas ribonucleo-proteínas nucleares (snRNP), que eliminan los intrones y unen las secuencias de ARNm codificantes, dando como resultado una molécula de ARNm funcional.

2.3.4. Codificación del código genético: codones

El genotipo codificado en cada gen del ADN consiste en una serie de instrucciones que sintetizan una proteína. Cada proteína queda determinada por una secuencia concreta de aminoácidos. El ARNm almacena esta información agrupando los ribonucleótidos en tripletes, llamados codones, lo que le permite codificar hasta 64 aminoácidos diferentes. Además de codificar un aminoácido, ciertos codones poseen otras funciones; por ejemplo, el codón AUG codifica la metionina y además, actuá de punto de inicio para la traducción genética.

2.3.5. Traducción del ARNm: síntesis proteica

El ARNm es transportado a los ribosomas (en el caso de eucariotas, atravesando los poros del núcleo celular hasta llegar al citoplasma) donde se lleva a cabo su traducción. El ribosoma es un orgánulo celular compuesto por un armazón de ácido ribonucleico, cono-

cido como ARN ribosomal (rARN) y por proteínas que se ensamblan en dos estructuras: subunidad mayor y subunidad menor. Su función es sintetizar las proteínas codificadas por el ARNm usando un adaptador: otro ácido ribonucleico presente en el citoplasma, conocido como ARN transferente (tARN). El tARN es una cadena (de alrededor de 80 ribonucleótidos) que aparece plegada con algunos de sus nucleótidos apareados, formando una estructura concreta, en la que se diferencia un triplete desapareado denominado anticodón que se apareará con su codón complementario de la secuencia de ARNm. En el otro extremo del tARN, los terminales 3' y 5' se pliegan formando un brazo aceptor con un grupo carboxilo que puede ligarse a un aminoácido concreto, en una reacción catalizada por unas enzimas específicas denominadas aminoasil-tARN-sintetasas. Además, aparecen dos bucles (o brazos) laterales: el bucle T, que reconoce al ribosoma y lo une al tARN durante el proceso de síntesis; y el bucle D, que identifica el tARN ante la enzima aminoasil-tARN-sintetasas, la cual unirá mediante un enlace covalente un aminoácido determinado al brazo aceptor del tARN. El proceso de síntesis comienza con las subunidades ribosomales desacopladas. La subunidad menor, posee tres sitios: sitio-A es el lugar de entrada del aminoacil- tARN, el sitio-P es el lugar en el que se produce el ensamblado del aminoacil-tARN con la cadena polipéptida (formando el complejo peptidil-tARN), el sitio-E por donde saldrá el tARN tras ceder su aminoácido. Se puede distinguir tres fases:

- Iniciación

  El primer paso es la adición de una serie de factores de iniciación (FI) a la subunidad menor, que bloquean el sitio-A, el sitio-E y facilita el acoplo del tARN de iniciación, que en células eucariotas es el anticodón UAC que sintetiza la metionina pero en procariotas es una variación de éste que sintetiza la formilmetionina (fmet-tARN); este aminoácido inicial suele eliminarse de la cadena proteica tras su formación. Posteriormente, la subunidad menor engancha el extremo 5' de la cadena de ARNm (con ayuda del CAP, en eucariotas) y comienza a desplazarse hasta encontrar el codón de iniciación AUG (en organismos procariotas la secuencia de reconocimiento está localizada a pocos aminoácidos del enganche y se conoce como secuencia de Shine-Dalgarno: AGGAGG). Una vez localizado, los factores de iniciación son liberados, permitiendo el acoplamiento de la subunidad mayor ribosómica y liberando el sitio-A y el sitio-E.

- Elongación

  El sitio-P se encuentra ocupado por un peptidil-tARN (que en un primer momento será el tARN de iniciación). El sitio-A está esperando a ser ocupado por un aminoacil-tARN que se apareará complementariamente al codón del ARNm. Una vez formado el nuevo par codón-anticodón en el sitio-A, la cadena polipéptida del sistio-P posee un nuevo aminoácido para su crecimiento; el polipéptido es transferido al sitio-A (reacción catalizada por la peptidil-transferasa, una ribozima del rARN) y el tARN del sitio-P queda descargado. Finalmente el ribosoma se desplaza un codón, trasladando el tARN descargado al sitio-E y el nuevo petidil-tARN ocupa el sitio-P. Este proceso se repite contínuamente, aumentando la cadena polipéptida, hasta que se alcance un codón de parada.

- Terminación

  Cuando se alcanza un codón de terminación, determinados factores de terminación actuan bloqueando el sitio-A y provocan la hidrólisis del peptil-tARN, liberando la proteína del complejo ribosoma-ARNm-tARN.

2.4. Microarrays

En los últimos veinte años, la técnica del microarray se ha consolidado como un método de análisis fundamental de la Genómica. Esta tecnología permite cuantificar simultáneamente númerosas interacciones biológicas a nivel molecular. Por ejemplo: (i) microarrays de expresión genética (chips de ADN), interacciones gen- proteína (ChIP-on-chip), (ii) variaciones en el número de copias cromosómicas (CGH arrays), (iii) polimorfismos de nucleótido simple (SNP arrays) o (iv) el estado de metilación del ADN (arrays de metilación). Un microarray consiste en una colección de biomoléculas activas, denominadas sondas, ordenadas e inmovilizadas sobre un sustrato sólido en regiones micrométricas denominadas spots. El material sobre el que se fijan las sondas puede ser muy variado. En los chips porosos, las sondas se adhieren mediante enlaces covalentes a pequeñas porciones de geles, membranas de nylon o nitrocelulosa depositadas sobre un portaobjetos de cristal. Por otro lado, en los arrays no porosos se fijan directamente a la superficie mediante enlaces covalentes. En este caso la superficie usada suele ser un sustrato de silicio, plástico,

oro o un recubrimiento de agarosa. Estas superficies sobre las que se depositan las sondas forman matrices bidimensionales molécularmente activas [M. López and Vega., 2002]. La técnica del microarray es un procedimiento estándar que permite detectar, cuantificar y analizar simultáneamente grandes cantidades de material biológico. Las biomoléculas contenidas en una muestra de interés se denominan dianas. La detección de estas dianas requiere un marcaje previo del material sometido a análisis. Este marcaje se consigue mediante diferentes indicadores luminiscentes, radiactivos o enzimáticos. Los métodos más usados son los enzimáticos y luminiscentes, en los que se utilizan tintes fluoróforos. El material marcado se distribuye sobre la superficie activa del microarray, donde cada diana se alinea complementariamente a una sonda específica, en un proceso denominado hibridación. El postprocesado de este microarray permite detectar e identificar las sondas hibridadas, se puede observar en la figura 2.2.

2.4.1. Cuantificación de la expresión genética: chips de ADN

En un microarray de expresión o chip de ADN, el material biológico considerado es una colección de moléculas de ADN de una sola hebra. Estas sondas pueden ser: (i) secuencias cortas (entre 20 y 100 nucleótidos) denominadas oligonucleótidos o (ii) fragmentos constituidos por varios miles de bases que forman genes o fragmentos de ellos. Cada sonda puede extraerse directamente de una muestra a partir de la clonación de ADN, procedente de librerías genéticas. La fabricación de chips de ADN se realiza de forma automática, en un proceso robotizado, con una duración inferior a las 16 horas. Una vez construido el microarray, es necesario obtener el material genético de estudio. En el caso de arrays de expresión, el material de interés es el ARNm transcrito. Una vez extraído el ARNm, se sintetiza su ADN complementario (cADN) con ayuda de una enzima denominada retrotranscriptasa. Esta enzima sintetiza un complejo cADN-ARN a partir de una hebra de ARN, proceso inverso al producido por la ARNp. Posteriormente, este complejo se desnaturaliza y se separan las dos hebras, obteniendo el cADN del ADN transcrito y libre de partes no codificantes. Las hebras de cADN se marcan y se colocan sobre la superficie del microarray donde se híbrida con su ADN sonda complementario. Finalmente, el microarray se lava y se elimina el resto de cADN diana no fijado. El cADN hibridado se detecta en un proceso acorde con el marcaje del material genético. En el caso de indi-

Figura 2.2: Preparación de un microarray de expresión. Cada portaobjeto contiene 4 microarrays, cada uno con 44000 spots cubriendo el genoma humano al completo. Tras ser extraído y marcado, el material genético se deposita sobre los arrays. Posteriormente, se cubre y se sella en una cámara de hibridación siguiendo el protocolo indicado. Posteriormente, tras retirar el banco de trabajo, se elimina el material genético sobrante y el chip queda listo para ser escaneado y procesado, figura adoptada de [Castillo, 2012].

cadores fluoróforos, el revelado consiste en detectar la fluorescencia de las dianas fijadas en el microarray, cuando este es iluminado con determinada longitud de onda. Este tipo de marcaje posee una ventaja: usando diferentes indicadores, con diferentes respuestas espectrales, se pueden analizar varias muestras en un mismo microarray. Por lo general, uno de los genomas analizados se utiliza como muestra de control, mientras que la otra es una muestra experimental en la cual se quiere observar una expresión diferencial respecto a la de referencia. Unos indicadores usados tradicionalmente son las cianinas Cy3 y Cy5 que emiten luz verde y roja cuando se iluminan con longitudes de onda λCy3 $\approx$ 550 (nm) y λCy5 $\approx$ 660 (nm) respectivamente [DeRisi et al., 1997].

El análisis de la intensidad de las dianas envuelve una fase de procesamiento de la ima-

gen del *microarray* para extraer la señal (expresión), analizar su variabilidad y evaluar la cualidad [Datta and Dougherty, 2009]. El objetivo final de esta fase es extraer la señal de cada diana y generar una matriz numérica conteniendo las intensidades de las señales. En general, las lineas de esta matriz corresponden a los genes, y las columnas a los experimentos de *microarray*.

De tal manera un experimento de *microarray* podría medir perfectamente el nivel de expresión de los genes en estudio. En la practica, sabemos que existe una variación en los datos obtenidos, y generalmente asumimos que pueden existir diversas fuentes de ruido: ruido en la preparación de la muestra, en el etiquetado, en la hibridación, en la extracción del brillo fosforescente de la diana, entre otros.

La hipótesis tras el análisis de datos del *microarray* es que cada lamina contiene, para cada gen, una representación del nivel de expresión de los genes. En general, un estudio envuelve datos de *microarray* los cuales intentan comparar las expresiones genéticas en *microarrays* distintos. De esta manera, los datos precisan pasar por un proceso de normalización antes de ser comparados. Uno de los motivos para que la normalización sea hecha es que la cantidad de ARN pueda ser diferente. Otro motivo es que pueda existir un sesgo en la etiqueta fosforescente.

Los datos de expresión genética provenido por *microarrays* pueden ser básicamente de dos tipos: *estacionario* o *temporales*. En los datos estacionarios, una muestra (*microarray*) no posee una relación de tiempo con las otra. Por ejemplo, un experimento en que varios *microarrays* son obtenidos, uno de cada paciente con cáncer. Por otro lado, los datos temporales representan un análisis de las expresiones a lo largo del tiempo. Por ejemplo, datos obtenidos a cada hora del ciclo de vida del *Plasmodium falciparum* (uno de los agentes que causa la malaria).

2.4.2. Normalización y Cuantización

La cuantización es una tarea de pre-procesamiento obligatorio para la estimación de la PGN. Los datos de expresión del *microarray* son valores reales resultantes de la aplicación de dos expresiones (Cy5/Cy3). Las etapas de normalización y cuantización envueltas en los experimentos de este capítulo, son aplicadas en el siguiente orden:

1. Aplicación de logaritmo en base 2 de todas las expresiones originales, resultando en la matriz **G**;

2. Las señales de **G** fueron normalizadas por una transformación normal dada por, para todo gen $g(t) \in G$, $\eta[g(t)] = \frac{g(t)-E[g(t)]}{\sigma[g(t)]}$, en que $E[g(t)]$ y $\sigma[g(t)]$ son, respectivamente, la esperanza y el desvío patrón de $g(t)$;

3. Sea $g'(t) \in \eta[g(t)]$. La cuantización de un gen g' es realizada por un mapeo definido, para todo t por:

$$g''(t) = \begin{cases} -1, & \text{si } g'(t) \geq h \\ 0, & \text{si } l \leq g'(t) \leq h \\ 1, & \text{si } g'(t) \leq l \end{cases} \tag{2.1}$$

   en que l es la media de los valores negativos de $g'(t)$ y h es la media de los valores positivos de $g'(t)$.

La transformación normal hace que todas las señales tengan media cero y desvío patrón 1, disponiendo todas las señales en una misma escala de variación. Aunque la transformación normal crea clases de equivalencia que disminuyen el error de estimación, ella amplifica el ruido en genes con perfiles de expresión casi constantes (*housekeeping*). Para usar la transformación normal, estos genes deben ser filtrados previamente.

La cuantización tiene el efecto de crear clases de equivalencia entre señales, disminuyendo así los errores de estimación debido a la falta de muestras. Esta es basada en que el hecho de la transformación normal hace que la media de cada señal tenga valor cero. De esta forma, el número de valores positivos y negativos serán aproximadamente el mismo. Es así, que los valores de sub-expresión (-1) estaban abajo de la media de los valores negativos y los valores de super-expresión (+1) estaban encima de la media de los valores positivos. Los valores próximos a la media general (encima de la media de los negativos y al mismo tiempo abajo de la media de los positivos) fueron considerados como valores normales (0).

2.5. Redes Complejas

El inicio de la teoría de grafos es atribuida a Leonard Euler que en 1736, probo la inexistencia de un camino que pase por todos los 7 puentes de Königsberg en un solo intento. Este problema fue resuelto utilizando un grafo, en el cual cada puente es una arista y los vértices representan las regiones conectadas por los puentes.

La teoría de redes complejas extiende el formalismo de la teoría de grafos por acrecentar medidas y métodos fundamentales en propiedades reales de un sistema. Esta extensión se concentra principalmente en la interpretación de que el objetivo de las redes es la representación de sistemas reales, por medio del análisis de datos experimentales, considerando que las redes son dinámicas, pudiendo modificar su estructura a medida que pasa el tiempo.

El primer modelo de redes complejas fue el de las redes aleatorias propuesta por Paul Erdös y Alfréd Rényi en 1959[Erdös and Rényi, 1959]. Desde entonces otros modelos de redes complejas fueron propuestas para la representación de sistemas reales, con destaque en los modelos: mundo pequeño (del inglés: *small-world*)[Watts and Strogatz, 1998] libre escala (del inglés: *scale-free*) [Barabási, 2009].

Los modelos de las redes complejas representan distintas topologías y propiedades bien definidas, las cuales pueden ser usadas para representar las GRNs, las cuales incluso pueden ser caracterizadas en términos de medidas de redes complejas como lo muestra [Costa et al., 2007]. De esta forma, la teoría de redes complejas permite la caracterización, análisis y representación de los mas variados sistemas complejos, como por ejemplo los sistemas biológicos como lo muestra [Kauffman, 1969, Kauffman, 1993, N. Guelzim and Képés, 2002, Albert, 2005, Costa et al., 2008].

Una red compleja es representada por un grafo $G = (V, E)$ compuesto por un conjunto $V = v_1, v_2, .., v_n$ de vértices (genes), conectados por un conjunto $E = e_1, e_2, .., e_n$ de aristas. Una red compleja posee un tamaño definido por el número de vértices n de la red y un grado medio $\langle k \rangle$ de aristas conectadas a los vértices.

A continuación mostramos los principales modelos de las redes complejas y sus propiedades.

2.5.1. Redes Aleatorias

Las redes aleatorias propuestas por Erdös y Rényi en su primer articulo [Erdös and Rényi, 1959] pueden ser consideradas el modelo mas elemental de las redes complejas. La arquitectura (topología) ER se basa en la relación aleatoria de los vértices considerando una distribución uniforme de probabilidades entre ellos. Este modelo de generación de redes se inicia con n vértices desconectados, y su topologia es definida por la inclusión aleatoria de m aristas entre los vértices, evitando el auto-relacionamiento y las conexiones múltiples. El nombre de la red aleatoria se refiere a la naturaleza desordenada de la organización de las aristas entre los vértices.

El modelo de Erdös y Rényi (ER) define n vértices y una probabilidad $0 < p < 1$ de conectar cada par de vértices de la red.

En este modelo en redes con gran tamaño $n \to \infty$, el número medio de conexiones $\langle k \rangle$ (grado medio) para cada vértice es dado por $\langle k \rangle = p(n-1)$.

A fin de construir redes ER y garantizar grados medios k semejantes entre sus vértices puede ser adoptado una probabilidad fija P de una artista entre dos vértices v_i y v_j , tal que:

$$P(v_i \leftrightarrow v_j) = \frac{\langle k \rangle}{n-1} \tag{2.2}$$

La distribución del número de conexiones por vértice $P(k)$ es bien aproximada por una distribución de Poisson, dada por:

$$P(k) = e^{-\langle k \rangle} \frac{\langle k \rangle^k}{k!} \tag{2.3}$$

En este sentido las redes ER, son llamadas también grafos aleatorios de Poisson (del inglés: *Poisson random graphs*).

2.5.2. Redes Mundo Pequeño

Las redes mundo pequeño (del inglés: *small-world*) fueron propuestas por Watts y Strogatz en 1988 [Watts and Strogatz, 1998]. Este modelo representa una alternativa al modelo aleatorio, asumiendo como hipótesis que las redes biológicas, tecnológicas y sociales pueden representar una topología que no es totalmente aleatoria.

Este modelo fue llamado *small-world* por analogía al fenómeno mundo pequeño, en el cual el investigador Stanley Milgran en 1967 descubrió que la distancia media (medida en términos de conexiones de conocimiento) entre dos personas en los Estados Unidos era próxima de seis. Este descubrimiento quedo conocida como *seis grados de separación*.
Con el objetivo de generar un modelo que no fuese totalmente aleatorio, fue considerado una red en forma de anillo conteniendo n vértices y k aristas por vértices, conectadas a sus vecinos mas próximos. Después de la construcción, cada arista puede ser reconectada con probabilidad p, permitiendo que la red sea ajustada entre una red regular (p) o aleatoria p y de esta manera permitir la generación de la topología intermediaria $0 < p < 1$.
Las redes WS son caracterizadas a partir de dos medidas principales: el tamaño de camino $L(p)$ y el coeficiente de agrupamientos $C(p)$. El $L(p)$ es definido como el número de aristas recorrido por el camino mas corto entre dos vértices, siendo calculadas la media sobre todos los pares de vértices (propiedad global). El $C(p)$ mide la conectividad de un vértice (propiedad local).
En general las redes WS representan la propiedad *small-world*, haciendo que la mayoría de los vértices puedan ser alcanzados por los demás vértices recorriendo un número pequeño de aristas. Otra propiedad de las redes WS es la presencia de un gran número de ciclos (*loops*) de tamaño tres, o sea, si un vértice v_i esta conectada a los vértices v_j y v_l, entonces la probabilidad de los vértices v_j y v_l de estar conectados es alta (el coeficiente de agrupamiento es alto)[Costa et al., 2007]. Las redes ER tienen la probabilidad del mundo pequeño, pero con un bajo coeficiente de agrupamiento.

2.5.3. Redes de Libre Escala

Los modelos ER y WS representan un patrón de conexiones aleatorias conteniendo un número de conexiones k similar entre sus vértices. Barabasi y Albert[Barabási, 2009], buscan entender la dinámica y la estabilidad topológica de las grandes redes reales, percibieron que independientemente del sistema analizado, la probabilidad $P(k)$ de un vértice de la red interacciona con k otros vértices el cual resulta como una ley de potencia, de la forma:

$$P(k) \sim k^{-\gamma} \tag{2.4}$$

en el cual el parámetro γ es una constante que determina el decaimiento exponencial, el cual puedes ser observado.

Las redes de libre escala (del inglés: *scale-free*) no representan una distribución homogénea de conexiones k entre sus vértices, representando pocos vértices altamente conectados a otros vértices de la red, y un gran número de vértices con pocas conexiones [Costa et al., 2007]. Estos vértices altamente conectados son llamados de *hubs*.

El modelo de red propuesto por Barabasi y Albert [Barabási, 2009](BA), es basado en dos reglas: *crecimiento y preferencia lineal de conexión*. La generación de redes BA es iniciada con la inclusión de $n_0 < n$ vértices conectados aleatoriamente, en general usando el modelo ER.

En la etapa de crecimiento de la red, a cada instante de tiempo $t = 1, 2, 3, 4, ..n - n_0$ un nuevo vértice v_i conteniendo $k < n_0$ aristas es adicionado en la red, siguiendo una preferencia lineal de conexión. O sea, la probabilidad de un vértice v_j ya existente en la red será conectado al nuevo vértice v_i, es linealmente proporcional al grado k_j del vértice v_j, tal que:

$$P(v_i \leftrightarrow v_j) = \frac{k_j}{\sum_u k_u}, \forall v_u \in V \tag{2.5}$$

Dado que todo nuevo vértice posee k aristas, la red en el tiempo t será $n = n_0 + t$ vértices y $m = kt$ aristas, correspondientes a una conexión media $k = 2k$, considerando un grafo con aristas no direccionadas. La preferencia de conexión para los vértices mas conectados también es conocida como paradigma **el rico queda mas rico**.

El modelo de redes *free-scale* y sus propiedades están siendo utilizadas para simular y describir el comportamiento de las GRNs.

2.6. Modelado y Simulación de las Redes de Regulación Genética

El dogma central de la biología molecular explica el proceso de la expresión genética mediante un mecanismo de regulación a nivel transcripcional. Sin embargo, el metabolismo celular de un organismo depende de numerosos procesos subyacentes, entre ellos la expresión genética, que interaccionan entre si de forma compleja para producir respuestas

diferentes. Por ejemplo: (i) la transducción de señales extracelulares que modifican la función de ciertas proteínas, alterando el proceso de regulación o (ii) modificaciones del estado de metilación del ADN que, sin producir mutaciones, alteran su estructura espacial e impiden la formación del complejo de preiniciación transcripcional e incluso (iii) la actividad de un gen puede estar controlada por la expresión de otros que codifican la síntesis de proteínas funcionales para su transcripción. El fenómeno de expresión genética es, por tanto, un proceso dinámico y complejo que integra diferentes elementos reguladores, internos y externos al sistema biológico considerado. Este mecanismo de adaptación funcional del proceso de expresión es lo que se conoce como epigenética [A. J. Hartemink and Young, 2000]. Un modelo que trata de simplificar los mecanismos de regulación desde un punto de vista fenomenológico, a niveles de interacción gen-gen, es la red de regulación genética (del inglés: *Genetic Regulatory Network* - GRN) [S. Das and Welch, 2008]. Este modelo proyecta todos los procesos reguladores, a diferentes niveles metabólicos, en el espacio de la actividad genética. Las GRN establecen relaciones causales directas entre un conjunto de genes, para ser exactos, entre su estado de expresión [A. de la Fuente et al., 2002]. En una GRN se considera que la expresión de un gen, al que se denomina hijo, depende del estado de expresión y la actividad de otros, denominados padres [Huang et al., 2009]. Las GRN heredan características propias de la teoría de grafos así como sus propiedades [A. Ribeiro and Kauffman, 2006]. Gráficamente, una GRN se caracteriza mediante un conjunto de nodos que representan los genes. Por otro lado, la estructura topológica de la red representa las relaciones causales entre los genes. En la Figura 2.3 se representa un ejemplo de una GRN. Cada gen está representado por un nodo mientras que las relaciones de R se representan con vértices dirigidos, desde los padres a sus hijos. Además, este esquema permite representar el efecto regulador de *activación* o *inhibición*. Cualitativamente, las GRN permiten analizar ciertas propiedades del genoma, tales como su funcionalidad y evolución; información de vital interés en la industria famacológica y en la investigación médica. [M. Sanchez-Castillo and Huang, 2011] [Mallat., 2009] [Huang et al., 2009].

En general los sistemas de regulación genética envuelven muchos genes conectados por interacciones positivas (Activación) o negativas (Negativas). Los métodos formales pueden ser aplicados para representar la estructura y el comportamiento de los sistemas

regulatórios. En especial los métodos de modelado permiten el análisis de los sistemas de regulación genética grandes y complejos. Otro aspecto importante es que estos modelos también permiten la simulación de una gran variedad de condiciones experimentales [Jong, 2002].

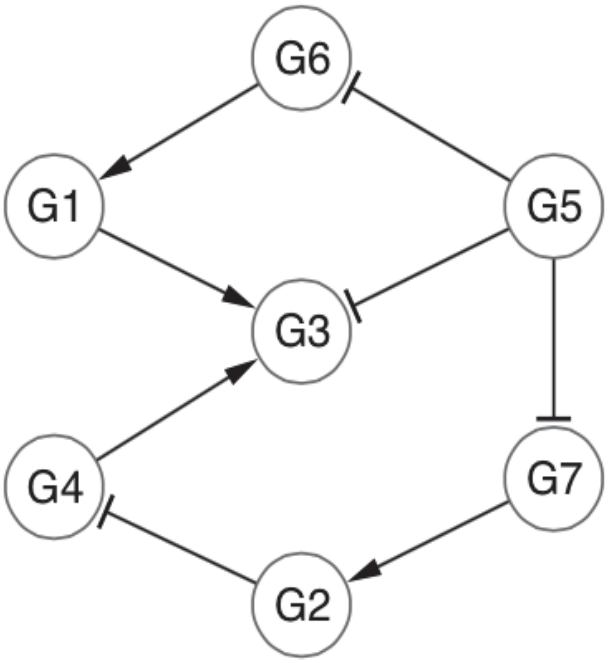


Figura 2.3: Representación gráfica de una GRN con G = 7 genes. Los nodos representan genes y los vértices las relaciones de parentesco entre ellos. El tipo de regulación se expresa con una punta en forma de flecha y con un extremo en forma de $\ll T \gg$, para efectos de activación e inhibición respectivamente. Por ejemplo, el gen G3 posee tres padres: G1 y G4 que activan su expresión y G5 que la inhibe. Por otro lado, el gen G1 es hijo de G6 que activa su expresión, figura adoptada de [Castillo, 2012].

Debido a la complejidad de los algoritmos, que actualmente existen en la literatura varios son los temas propuestos para el modelado e identificación de los GRNs. Hay dos enfoques principales para modelar matemáticamente las redes complejas de interacciones genéticas [Shmulevich and Dougherty, 2007]. Uno de los enfoques es considerar las variables en el dominio continuo, empleando ecuaciones diferenciales y sus variaciones para construir modelos cuantitativos detallados de redes bioquímicas con funcio-

nes celulares de interés [Jong, 2002]. El segundo enfoque se basa en la construcción de modelos cualitativos discretos de interacción genética, incluyendo las Redes Booleanas [Kauffman, 1969], Redes Booleanas Probabilísticas [Shmulevich et al., 2002] y las Redes Genéticas Probabilísticas [Barrera et al., 2007]. Aunque el enfoque continua ofreciendo una comprensión detallada del sistema en cuestión, en general hace falta un considerable conjunto de muestras además de informaciones sobre determinadas características de las reacciones [Karlebach and Shamir, 2008], que hace conveniente en un número limitado de situaciones. Por otro lado, los enfoques discretos pueden ser fácilmente modeladas computacionalmente, habiendo sido empleado con éxito en el modelado y simulación de algunas redes y procesos biológicos, tales como *Drosophila melanogaster* [Sánchez and Thieffry, 2001, Albert and Othmer, 2003], ciclo celular de la levadura [Li et al., 2004, Zhang et al., 2006, Davidich and Bornholdt, 2008], *Arabdopsis thaliana* [Espinosa-Soto et al., 2004], *Saccharomyces cerevisiae* [Li and Lu, 2005], ciclo celular de mamíferos [Faure et al., 2006], *Plasmodium falciparum* [Barrera et al., 2007], entre otros.

En particular, las redes bayesianas son un modelo probabilístico capaz de representar una red genética causal. En este tipo de modelado, las entidades biológicas (genes, proteínas y otras moléculas) son representadas como nodos de un grafo que son conectados por aristas que indican un cierto relacionamiento entre ellos. Las redes bayesianas son ampliamente utilizadas para representar redes genéticas [Friedman et al., 2000, Kelemen et al., 2008]. Tal modelo utiliza distribución de probabilidades, teoría de los grafos y propiedad local de Markov (cada variable es condicionalmente independiente de sus no-ancestros) para representar las relaciones entre variables y estados con el objetivo de hacer inferencias.

El modelo de Redes Booleanas (del ingles: *Boolean Networks* - BNs) constituye un tipo de red bayesiana dinámica discreta que representa un modelo adecuado para generalizar y capturar el comportamiento de los sistemas biológicos a nivel global (cualitativo), teniendo en cuenta el limitado número de experimentos (muestras), de alta dimensionalidad de variables (genes) y de la naturaleza ruidosa de las medidas de expresión [Kauffman, 1969, Lähdesmäki et al., 2006]. Aunque son útiles las BNs en varios casos, una importante limitación es su determinismo, que hace la suposición de un entorno sin incertidumbre. Además de eso, es importante considerar una célula como un siste-

ma abierto, que puede recibir estímulos externos. Dependiendo de las condiciones externas en un determinado instante de tiempo en que la célula puede cambiar su dinámica [Shmulevich and Dougherty, 2007]. Para lidiar con este problema es propuesto un tipo especial de BNs, las Redes Booleanas Probabilísticas (PBNs), en el cual además de considerar los genes con valores binarios, es asociado a cada uno de ellos un conjunto de funciones booleanas predictoras, asignando una probabilidad a cada función específica [Shmulevich et al., 2002]. Aunque este enfoque también tiene desventajas importantea, la desventaja más acentuada es la pérdida de información debido a la discretización de los datos. Pero eso hace que los modelos booleanos sean tratados más simples y fáciles computacionalmente. [Styczynski and Stephanopoulos, 2005]. Una discusión al respecto de esto puede ser vista en [Ivanov and Dougherty, 2006]. Este trabajo adopta los modelos de redes booleanas y redes booleanas probabilísticas para el modelado de las redes genéticas probabilísticas, ya que ellas son capaces de capturar las propiedades globales de las GRNs con cantidades limitadas de muestras.

Las próximas subsecciones muestran algunas alternativas para el modelado y simulación de las redes de regulación genética, en las cuales las expresiones de los genes son representadas por valores discretos.

2.6.1. Modelos Basado en Grafos

Inferir redes a partir de datos de expresión genética constituye uno de los problemas mas desafiadores de la bioinformática [Kelemen et al., 2008]. Los genes pueden ser visto como nodos de una red compleja, los cuales poseen como entrada las proteínas, etc, los factores de transición y tienen como salida la cantidad de transcritos.

El uso de grafos es utilizado para la representación y modelado de redes de regulación genética el cual permiten capturar el comportamiento colectivo de los genes que constituyen una red y el relacionamiento entre ellos de forma directa, y su topología exhibe todos estos relacionamiento de forma explicita, el cual se mantienen independiente de la atribución de algún valor numérico en los genes o en sus relaciones. Por esta razón los grafos son ampliamente utilizados para denotar una relación de dependencia.

Las propiedades estructurales de los grafos, pueden ser muy útiles para la predicción de relacionamientos funcionales en redes biológicas y también para la caracterización de los

efectos de perturbación en elementos de esta red.

Un grafo dirigido (digrafo) G es definido como una tupla (V, E), siendo V el conjunto de vértices, y $E \subseteq VxV$ un conjunto de aristas. Una arista dirigida $e_{i,j}$ es un par ordenado de vértices que deja el vértice v_i y llega al vértice v_j. Este relación puede ser representado por una matriz de adyacencia $M = (e_{i,j})$ correspondiente al grafo $G = (V, E)$, en el cual

$$e_{i,j} = \begin{cases} 1, & \text{si}(i,j) \in E \\ 0, & \text{caso contrario} \end{cases}$$

Y los vértices son enumerados consecutivamente $1, 2, .., |V|$.

Considerando que un dígrafo represente relacionamientos regulatorios entre los genes, entonces la artista $e_{i,j}$ puede ser usada para representar una relación entre el "factor de transición" v_i activando un gen v_j. La autoregulación corresponde a un lazo del grafo, o sea es una arista $e_{i,j}$.

En muchos casos la direccionalidad de estas aristas no es importante, por ejemplo en interacciones entre las proteínas [Dougherty, 2011]. En este contexto pueden ser usados grafos no dirigidos, o simplemente grafos, en los cuales el conjunto de aristas E son pares no ordenados de vértices y la matriz de adyacencias M es simétrica ($e_{i,j} = e_{j,i}$) y antireflexiva (no posee lazos).

Los vértices de un dígrafo poseen algunas propiedades, como el grado de entrada o el grado de salida. El grado de entrada de un vértice v_i es el número de aristas incidentes al vértice v_i (artistas recibidas de otros vértices de red). De forma similar, el grado de salida es el número de aristas que dejan el vértices v_i e inciden en otros vértices de red. Estas medidas representan cuantos vértices ïnfluencian.ªl vértice v_i y cuantos vértices son ïnfluenciados"por el vértice v_i, respectivamente. La figura 2.4 muestra un ejemplo de dígrafo (a) y su matriz de adyacencia(b).

La caracterización de las redes biológicas por medio de los grafos pueden generar informaciones relevantes para el conocimiento estructural y organizacional de estas redes.

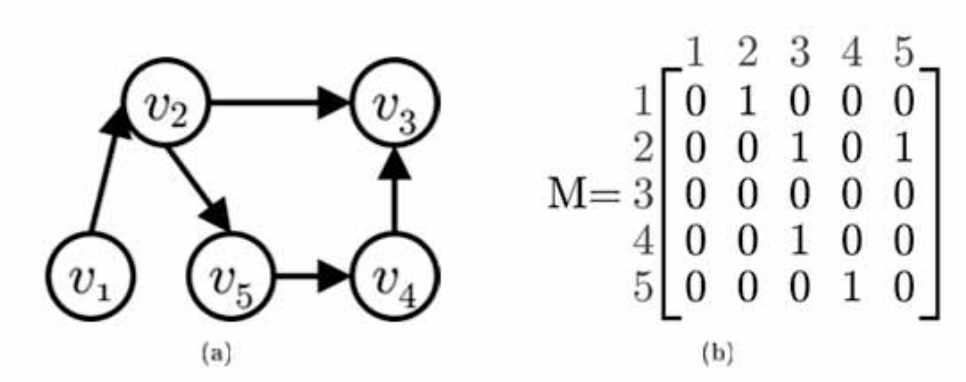


Figura 2.4: Ejemplo de un dígrafo con 5 vértices (a) es su matriz de adyacencia (b), Cada elemento igual a 1 en la matriz de adyacencia representa un relacionamiento regulatorio entre dos vértices del dígrafo, figura adoptada de [Lopes et al., 2011a]

2.6.2. Redes Booleanas

Las redes booleanas (del inglés: *Boolean Network* - BN) fueron introducidas por Kauffman [Kauffman, 1969] para la modelación de la dinámica de sistemas complejos y en particular, de la GRNs. Las BNs se definen por un conjunto de vértices $V = \{v_1, v_2, ..., v_n\}$ y un conjunto de funciones booleanas $\Phi = \{\phi_1, \phi_2, ..., \phi_n\}$, uno para cada gen, también conocidas como funciones de transición booleanas. [D'haeseleer et al., 1999].

Cada gen $v_i \in \{0, 1\}, i = 1, 2, ..., n$ representa una variable binaria, y su valor en el instante de tiempo $t + 1$ es completamente determinado por los valores de sus k genes predictores en el instante de tiempo t. Más formalmente, podemos representar esta dinámica como $v_i(t + 1) = f_i(v_{1i}(t), v_{2i}(t), ..., v_{ki}(t))$, en el cual $v_{1i}, v_{2i}, ..., v_{ki}$ representan los k genes predictores o genes reguladores que poseen aristas incidentes al gen v_i (objetivo).

Por lo tanto, las funciones booleanas Φ son usadas para actualizar los genes, considerando iteraciones discretas en el tiempo, siendo todos los genes actualizados de forma sincronizada de acuerdo con la función asignada a él. Este proceso sincrónico simplifica el cálculo y conserva las características generales de la dinámica de red [Kauffman, 1969].

En este tipo de red la dinámica es determinista, es decir, la elección de los k predictores y sus respectivas funciones lógicas para cada gen permanecen sin cambios durante todos los instantes de tiempo. Cuando las funciones booleanas ϕ_i son elegidas de forma aleatoria

por cada uno de los genes, la BN recibe el nombre de red booleana aleatoria. (del ingles: *Random Boolean Network*) [Shmulevich and Dougherty, 2007].

El estado de un gen v_i en una BN es definido por el valor asumido por él, $v_i = 1$ representa que el gen esta activo o $v_i = 0$ inactivo. Un estado $\vec{s}$ de una BN es definido por los valores de todos los genes en un determinado instante de tiempo, $\vec{s}(t) = (v_1, v_2, ..., v_n), v_i \in \{0, 1\} \forall i = 1, 2, ..., n$.

Para cada BN tenemos 2^n posibles estados definidos por $S = \{\vec{s}_1, \vec{s}_2, ..., \vec{s}_z\}, z = 2^n$, siendo n el número de genes de la red (BN). Pero no siempre todos los 2^n estados son observados (están presentes) en una red, mientras que ciertos estados serán frecuentemente observados, dependiendo del estado inicial elegido. Estos estados que son observados periódicamente conforman los llamados atractores (ciclos) y los estados que conducen hasta los atractores son llamados estados transitorios, los cuales conforman la cuenca de atracción representado por el atractor correspondiente.

Los atractores son estados estacionarios de un sistema dinámico, que capturan el comportamiento del sistema a largo plazo [Shmulevich and Dougherty, 2007]. Los atractores son siempre cíclicas y pueden estar formadas por uno o más estados. El número de estados que un sistema puede visitar antes de retornar a un estado que ya ha visitado es denominado tamaño del ciclo. Kauffman [Kauffman, 1969] interpreta que los atractores de un BN pueden ser vistos como diferentes tipos celulares, argumentando que diferentes células son caracterizadas por su patrón recurrente de la expresión genética, de cierta forma correspondiente a los atractores de una BN. Las GRNs reales son altamente estables en presencia de perturbaciones ocasionadas por factores externos, sea de cualquier gen aislado o de varios genes. Considerando el formalismo de las BNs, esto significa que, cuando un número mínimo de genes son perturbados, estos genes cambian de valores (estados), pero los estados de red siguen en la misma cuenca de atracción y eventualmente llegan al mismo atractor [Shmulevich and Dougherty, 2007, Li et al., 2004, Zhang et al., 2006]. Esta estabilidad de las redes reguladoras en los organismos vivos permite que las células mantengan su estado funcional en el organismo incluso cuando se somete a las perturbaciones externas.

2.6.3. Redes Bayesianas

Los modelos estocásticos de las redes genéticas difieren de los modelos deterministicos por la incorporación aleatoria o incerteza. Las BNs asumen una dependencia funcional fija entre sus elementos, por eso son clasificados como modelos deterministicos de representación de las GRNs. Los modelos estocásticos difieren de los modelos deterministicos por incorporar aleatoridad o incerteza. Los valores asumidos por los genes de estos modelos pueden ser descritos por la distribución de probabilidades. El modelo estocástico más usado para representar las GRNs son las Redes Bayesianas (del inglés: *Bayesian Network* - BN) [Friedman et al., 2000, Kelemen et al., 2008].

La estructura de una red genética por medio de una red bayesiana es modelada por un grafo aciclico dirigido (del inglés: *Directed Acyclic Graph* - DAG) donde $G = (V, E)$. Los vértices $v_i \in V, 1 \leq i \leq n$, representan los genes y corresponden a las variables aleatorias X_i. Considerando que v_i es un gen, entonces X_i describe el nivel de expresión del gen v_i. Para cada X_i, existe una distribución de probabilidad condicional *P(X_i | predictores (X_i))* es definida, en la cual los *predictores(X_i)* representan las variables que regulan directamente al gen v_i de la red G.

El modelo de las redes Bayesianas constituyen dos partes: la parte cualitativa, en la cual las influencias directas (causal) entre dos genes son descritas por las aristas direccionales E en un grafo G; y la parte cualitativa, en la cual las distribuciones condicionales *p(X_i | predictores (X_i))*, representan los niveles de expresión, estas son anexadas a los genes de esta red. Luego los niveles de expresión X_i de un gen v_i son considerados como variables aleatorias y las aristas representan dependencias condicionales entre las distribuciones de estas variables aleatorias [Hovatta, 2005].

Las redes bayesianas incorporan una hipótesis Markoviana, en la cual determina que cada variable X_i es independiente de los otros genes (no predictores) dados sus predictores en G [Dougherty, 2011]. Una distribución conjunta que satisfaga esta hipótesis Markoviana puede ser descompuesta en un producto de las probabilidades condicionales dado por:

$$P(X_1, .., X_n) = \prod_{i=1}^{n} P(X_i \mid predictores(X_i)) \tag{2.6}$$

La red bayesiana mostrada en la figura 2.5 posee 5 vértices. Considerando la hipótesis markoviana el vértice v_3 es independiente del vértices v_4 dados los vértices v_1 y v_2, o sea, P(v_3 | v_1,v_2,v_4) = P(v_3 | v_1,v_2). De forma similar P(v_4 | v_1,v_2,v_3) = P(v_4 | v_2), dado que v_1,v_3 y v_5 no están conectados con al vértice v_4 y el vértice v_2 es su único predictor. La distribución conjunta de este ejemplo puede ser vista como:

$$P(v_1, v_2, v_3, v_4, v_5) = P(v_1)P(v_2)P(v_3|\, v_1, v_2)P(v_4|\, v_2)P(v_5|\, v_3) \tag{2.7}$$

De esta forma, la representación gráfica de una red Bayesiana exhibe la distribución

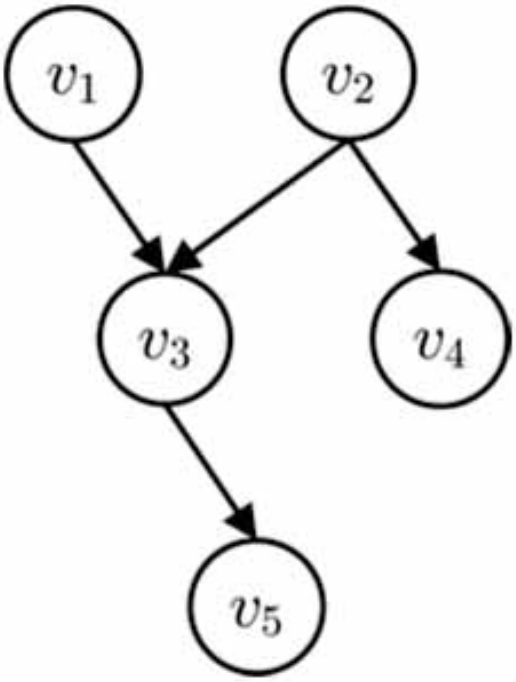


Figura 2.5: Ejemplo de una red Bayesiana conteniendo 5 vértices, figura adoptada de [Lopes et al., 2011a]

de probabilidad conjunta de una forma compacta e intuitiva [Dougherty, 2011]. Esta suposición de independencia permite la representación mas compacta de las dependencias condicionales entre los vértices. Se considera que el valor posible para cada vértice sea apenas uno de los posibles estados 0 ó 1 (activo o inactivo). En este caso, para representar la tabla de probabilidad condicional para el vértices v_3, será necesario $2^5 = 32$ entradas.

2.6.4. Redes Booleanas Probabilísticas

Una consideración muy importante es que la célula es un sistema abierto y no un sistema cerrado. En otras palabras, ellas pueden recibir estímulos externos. Dependiendo de las condiciones externas en un determinado instante de tiempo, la célula puede alterar su dinámica [Dougherty, 2011]. En este sentido las BNs representan un modelo cerrado, debido a que no consideran la estimulación externa. Por otro lado las Redes Booleanas Probabilísticas (del inglés: *Probabilistic Boolean Network* - PBN) se muestran como un modelo generalizado por representar las GRNs, permitiendo la inclusión de estímulos externos y sus efectos en la dinámica de un sistema. Conceptualmente esta es la principal diferencia entre las BNs y las PBNs.

La idea básica de las PBNs es usar varias funciones booleanas en conjunto, de forma que a cada instante de tiempo, una de ellas pueda ser escogida con una cierta probabilidad para determinar el valor del gen objetivo. Las PBNs pueden ser interpretadas como un conjunto de BNs que describen el funcionamiento de una GRN, haciendo que cada una de las BNs puedan ser escogidas para definir la dinámica de un sistema en un determinado instante de tiempo, las cuales representan condiciones o estímulos de este sistema.

De esta forma para cada gen v_i de una PBN es definido un conjunto de funciones booleanas $\Phi_i = \phi_j^{(i)}$, tal que $j = 1, .., l(i)$, en el cual cada $\phi_j^{(i)}$ es una posible función booleana que puede ser elegida para determinar el valor del gen v_i y $l(i)$ es el número de posibles funciones atribuidas al gen v_i. Se nota claro que si $l(i) = 1$ para todo $i = 1, ..., n$ entonces la PBN se reduce a una BN. En esta notación, cada función $\phi_j^{(i)}$ define el circuito lógico y también los predictores asociados al gen v_i.

La dinámica de una PBN en un determinado instante de tiempo es determinado por un vector de funciones booleanas, como en el caso de las BNs. La diferencia esta en la forma como estas funciones son elegidas, antes de forma determinística, ahora de forma probabilística. Considerando u posibles realizaciones de un vector de funciones booleanas $\Phi_1, \Phi_2, .., \Phi_u$ siendo definida por:

$$u = \prod_{i=1}^{n} l(i) \tag{2.8}$$

Luego es necesario definir las probabilidades de elección entre las funciones booleanas de cada gen v_i. De esta forma, la probabilidad de $\phi_j^{(i)}$ será elegida para determinar la

dinámica del gen v_i, tal que $(1 \leq j \leq l(i))$, en un determinado instante de tiempo es definida por $c_j^{(i)}$. Considerando que $c_j^{(i)}$ es una distribución de probabilidades, es necesario satisfacer la ecuación 2.9.

$$\sum_{j=1}^{l(i)} c_j^{(i)} = 1 \tag{2.9}$$

La PBN se dice que es independiente si la elección de las funciones booleanas $\phi_j^{(1)}$,$\phi_j^{(2)}$,..,$\phi_j^{(n)}$ fueran independientes [Dougherty, 2011].

La dinámica de las PBNs, como esta descrita aqui es esencialmente la misma de las BNs. Sin embargo en un determinado instante de tiempo, el estado de un gen v_i es definido por una de las $l(i)$ posibles funciones (predictores), de acuerdo con su respectiva probabilidad. Esto también puede ser visto como si en cada instante de tiempo fuese elegida una BN, de las u posibles, para determinar la dinámica de la red en este instante de tiempo.

Las PBNs representan una interfaz entre el absoluto determinismo de las BNs y la naturaleza probabilística de las Redes Bayesianas por incorporar incerteza en la elección de las funciones de transición booleanas, las cuales determinan la dinámica del sistema.

2.6.5. Redes Genéticas Probabilísticas

Las redes genéticas probabilísticas (del inglés: *Probabilistic Genetic Network* - PGN) fueron propuestas por [Barrera et al., 2007], como un modelo para representar las GRNs. Las PGNs están basadas en las PBNs, en las cuales la elección de la función de transición no es determinística y los estados de los genes y de la red son determinados por valores discretos.

Las PGNs pueden ser representadas como un sistema dinámico finito, discreto en el tiempo y con un número finito de estados, en el cual cada transición es representado por una variable que recibe el valor de expresión de este transcrito. La composición de todas estas variables forman un vector *llamado estado del sistema*. Donde cada componente de este vector posee un función asociada que calcula su próximo valor a partir del estado anterior de otros genes (predictores), siendo denominada *función de transición*, denotada por ϕ. Estas funciones son componentes de un vector de funciones de transición Φ, el cual define la transición de la red para el próximo estado y representa el mecanismo de la regulación genética [Barrera et al., 2007].

Sea R el conjunto de valores de todos los componentes, por ejemplo $R = 0, 1$ es representado como un sistema binario. El vector de funciones de transición $\Phi = \phi_1, \phi_2, .., \phi_n$ para una red conteniendo n genes, es una función de R^n en R^n. Un sistema dinámico es dado por [Barrera et al., 2007]:

$$s(t+1) = \phi(s(t)) \tag{2.10}$$

en que $s(t) \in R^n$, $t \leq 0$, representa el estado de la red en un determinado instante de tiempo t. Un componente de $s(t)$ es un vector $x_i(t) \in R$. Los sistemas definidos de esta forma son *invariantes a la traslación en el tiempo*, la función de transición es la misma para todo el tiempo discreto t. Cuando ϕ es una función estocástica, etc, para cada estado $s(t)$, el próximo estado $\phi(s(t))$ es una realización del vector aleatorio, el sistema dinámico es un proceso estocástico.

En las PGNs, las redes de expresión genética son representadas como un proceso estocástico, siendo la función estocástica un caso particular de la cadena de Markov Considere una secuencia de vectores aleatorios $S_0, S_1, S_2, ..$ asumiendo valores en R^n y sus realizaciones denotadas respectivamente por $s(0), s(1), s(3), ..$ Una secuencia de estados aleatorios $(S_t)_{t=0}$, es llamada cadena de Markov si para todo $t \leq 1, P(S(t) = s(t)|S(0) = s(0), .., S(t-1) = s(t-1)) = P(S(t) = s(t)|S(t-1) = s(t-1))$.

En otras palabras, asumir este principio significa que la probabilidad condicional de un evento futuro, es dado por los eventos anteriores, solo depende apenas del evento inmediatamente anterior. Una cadena de Markov es caracterizada por una *matriz de transición* $\pi_{y|x}$ de probabilidades condicionales entre los estados, siendo sus elementos denotados por $p_{y|x}$, es un vector de estados iniciales s_0. Una PGN es una cadena de Markov $(\pi_{Y|X}, s_0)$, en el cual son asumidos los siguientes axiomas:

- la matriz de transición $\pi_{Y|X}$ es homogénea, en otras palabras., $p_{y|x}$ no es una función de t. Las probabilidades de transición de estados son constantes a lo largo del tiempo.
- $p_{y|x} > 0$, en otras palabras. todos los pares de estados $x, y \in R^n$, pueden ser atendidos (cadena de Markov ergódica).
- la matriz de transición $\pi_{Y|X}$ es condicionalmente independiente , en otras palabras, para todo par de estados $x, y \in R^n, p_{y|x} = \prod_{i=1}^{n} p(y_i|x)$.

- $\pi_{Y|X}$ es casi deterministico., en otras palabras, para todo estado $x \in R^n$, existe un estado $y \in R^n$ tal que $p_{y|x} \approx 1$.

Estos axiomas que definen las PGNs son motivadas por fenómenos biológicos o simplificaciones debido a la falta de datos para la estimación del modelo, por ejemplo en los experimentos de *microarrays* temporales, en los cuales existen pocas observaciones a tiempo frente a millones de genes. El primer axioma es una restricción para simplificar el problema de estimación en la que podría ser fácilmente generalizado. El segundo axioma impone que todos los estados son alcanzables, o sea, asume que la presencia de ruido o perturbación pueda llevar el sistema a cualquier estado. El tercer axioma determina que la expresión de un gen en un determinado instante de tiempo t independiente de la expresión de otros genes en el mismo instante t. El cuarto axioma dice que el sistema tiene una dinámica estructural que esta sujeta a pequeños ruidos [Barrera et al., 2007].
Es importante observar que el tercer axioma puede no ser verificado dependiendo de la limitación de la resolución temporal de los datos experimentales disponibles. Sin embargo, este axioma fue adaptado en este modelo para permitir alguna tratabilidad estadística. Usando este axioma fue posible conseguir resultados biológicos bastante significativos en datos del *Plasmodium falciparum* según [Barrera et al., 2007] y también en la recuperación de redes a partir de datos simulados [Lopes et al., 2011a, Lopes et al., 2011b, Lopes et al., 2014].

2.7. Reconocimiento de Patrones y Selección de Características

En análisis de señales genómicos, se tiene una amplia variedad de problemas en los cuales implican el reconocimiento de patrones. Por ejemplo, los *microarrays* contienen medidas de expresión de miles de genes y uno de los principales objetivos es la clasificación de los patrones a partir de los perfiles de expresión. Por lo tanto, se requiere el diseño de un clasificador ψ que reciben como entrada un vector de niveles de expresión genética $\mathbf{X} = (x_1, x_2, ..., x_n)$ y devuelve una etiqueta que predice la clase $Y = \{0, 1, ..., c-1\}$ a la cual el vector considerado pertenece. Un problema típico en análisis de expresión genética

es la clasificación de los diferentes tipos de cáncer o diferentes etapas de desarrollo del tumor [Porter et al., 2001]. Los clasificadores están diseñados en base a un conjunto de muestras (vectores de expresión) que se pueden derivar de diferentes tejidos o de un mismo tejido, que en general está sujeta a varias condiciones o en diferentes etapas del ciclo celular.

El problema de la selección de características consiste en seleccionar un subconjunto de características que representa adecuadamente los objetos en estudio. Una técnica de selección de características es dividida en dos partes principales: un algoritmo de búsqueda y una función criterio [Theodoridis and Koutroumbas, 2006]. En el análisis de expresión genética, las características son los genes, cuyos valores están dados por la expresión genética. Los conjuntos de datos de expresión genética generalmente cuentan con miles de características. Algunos métodos de inferencia de GRNs que aplican técnicas de selección de características han sido propuestos en la literatura [Liang et al., 1998, Butte and Kohane, 2000, Hashimoto et al., 2004, Peng et al., 2005, Margolin et al., 2006, Faith et al., 2007, Barrera et al., 2007, Zhao et al., 2008, Dougherty et al., 2008].

2.7.1. Problema de la dimensionalidad

Un importante paso para el diseño de un sistema de clasificación es la evaluación de el desempeño de un clasificador, en el cual la probabilidad de error de una clasificación es estimada. Además de la complejidad computacional, otra motivación para la selección de características es la existencia del problema de la dimensionalidad, en la cual el error del clasificador esta en función del número de características que describen los patrones (dimensionalidad) el cual forma una "curva en U" (ver Figura 2.6) [Jain et al., 2000]. Observando esta figura, se certifica que para las dimensiones menores que d_1, la adición de características implica una mejora en el desempeño esperado del clasificador. Entre las dimensiones d_1 y d_2, la inclusión de características pasa a no causar cualquier impacto significativo en su desempeño. El problema de la dimensionalidad comienza tras ocurrir la partida d_2 en que las nuevas características pasan a afectar negativamente el desempeño esperado del clasificador.

El número de muestras necesarias para que un clasificador tenga un rendimiento sa-

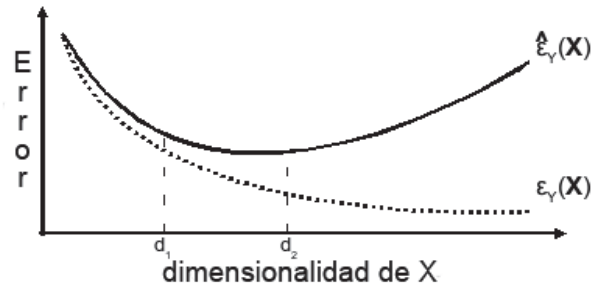


Figura 2.6: Gráfico de las tasas de error en función de la dimensionalidad con número fijo de muestras ilustrando el problema de la dimensionalidad. La curva del error Bayesiano (error del clasificador óptimo) es dado por $\varepsilon_Y(\mathbf{X})$, mientras que la curva de error esperado al aplicar un clasificador proyectado a partir de un número finito de muestras es dada por $\hat{\varepsilon}_Y(\mathbf{X})$, figura adoptada de [Martins Junior, 2009].

tisfactorio es exponencial con relación a la dimensión del vector de características [Jain et al., 2000]. Debido a esto, es muy común que un clasificador se muestre excesivamente ajustado a los datos de evaluación (*overfitting*) si el número de características seleccionado para el diseño del clasificador es mucho mayor en comparación con el tamaño del conjunto de muestras de evaluación.

2.7.2. Reticulados Booleanos

En la gran mayoría de los casos, y particularmente en el contexto de selección de características, los reticulados booleanos son estructuras algebraicas que normalmente representan el conjunto potencia (*power-set*) de un conjunto de elementos, o sea, todos los conjuntos propiamente contenidos en este conjunto, incluyendo el conjunto vacío y el conjunto total. En la selección de características, normalmente los elementos son las características de los objetos en estudios. De este modo, el reticulado booleano representa el espacio de búsqueda de todos los subconjuntos de características posibles. La Figura 2.7(a) ilustra un reticulado booleano representando todos los posibles subconjuntos del conjunto $\mathbf{X} = \{X_1, X_2, X_3\}$, de este modo la ilustración de la Figura 2.7(b) muestra las cadenas binarias correspondientes de los subconjuntos, en los cuales cada bit representa ausencia (0) o presencia (1) de uno de los elementos en un subconjunto dado. Por ejemplo, el subconjunto $\{X_1, X_3\}$ es representado por la cadena 101, ya que los elementos X_1 y X_3 están

presentes, mientras que el elemento X_2 está ausente del subconjunto de características en cuestión. Es importante destacar que las aristas del reticulado booleano representan la vecindad entre dos subconjuntos, de tal forma que dos subconjuntos son vecinos si la diferencia entre ellas es de apenas un elemento. En otras palabras, una arista representa un mapeo de un subconjunto a otro por la adición o sustracción de un elemento específico.

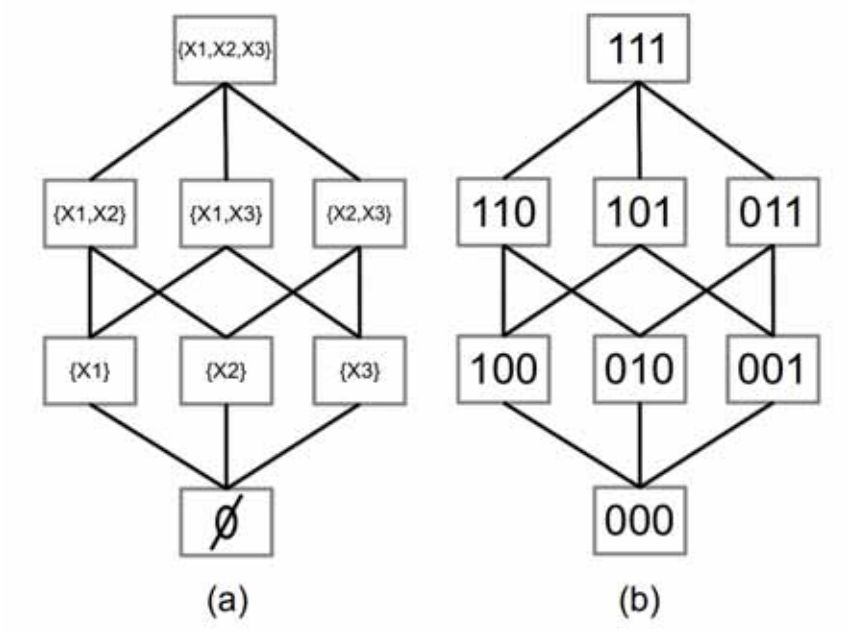


Figura 2.7: Reticulado Booleano de grado 3 representando todos los posibles subconjuntos de 3 elementos: (a) descripción de los subconjuntos; (b) cadenas binarias correspondientes, figura adoptada de [Montoya, 2014].

Los algoritmos de búsqueda en selección de características determinan un paseo a lo largo del reticulado booleano de modo que busca el subconjunto que optimice una determinada función criterio. Tal función criterio mapea cada subconjunto en un determinado valor que cuantifica su cualidad en representar los objetos en estudio.

En este trabajo, como el enfoque es en relación a la función criterio, los reticulados booleanos presentados en el Capítulo 3 para explicar los métodos desarrollados representan un concepto diferente de lo presentado hasta ahora. Tales reticulados representan todos los posibles valores (instancias o configuraciones) de un conjunto de elementos, en que cada elemento posee dos valores posibles en el modelo de redes booleanas para representación de las redes genéticas (ver Sección 2.6.2). Y las aristas de estos reticulados representan

una relación de vecindad entre las configuraciones de acuerdo con la distancia de HammingLorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.[2]. O sea, existe una arista entre dos configuraciones si y solamente si, dos configuraciones poseen distancia de Hamming igual a 1. Por ejemplo, en la Figura 2.7(b), el vértice 101 correspondiente a la configuración $\{X_1, X_2, X_3\} = \{1, 0, 1\}$. Es así que son configuraciones que difieren apenas por el estado de un gen.

2.7.3. Algoritmos de Búsqueda

Como discutido anteriormente, los algoritmos de selección de características recorren parte del conjunto potencia del conjunto total de características en busca de un subconjunto que optimice una determinada función costo. Hasta el momento, no se conoce un algoritmo polinomial para resolver el problema de la selección de características [Pudil et al., 1994, Somol et al., 1999, Nakariyakul and Casasent, 2009]. Consecuentemente, la búsqueda exhaustiva, la cual recorre todo el espacio de búsqueda, es el único algoritmo capaz de obtener la solución óptima en general, aunque existan algoritmos del tipo *branch-and-bound* que garanticen la optimibilidad para funciones criterio con estructuras especificas [Jain et al., 2000, Somol and Pudil, 2004, Ris et al., 2008]. La Figura 2.8 representa la taxonomía de los principales métodos utilizados en reconocimiento de patrones. Tales algoritmos son categorizados de acuerdo con tales dualidades como óptimo (devuelve la mejor solución) *versus* sub-optimo, determinístico (devuelve siempre

[2]La distancia de Hamming entre dos cadenas es el número de posiciones en las cuales ellas difieren entre si

la misma solución) *versus* estocástico (puede devolver soluciones diferentes en ejecuciones distintas), única solución *versus* varias soluciones.

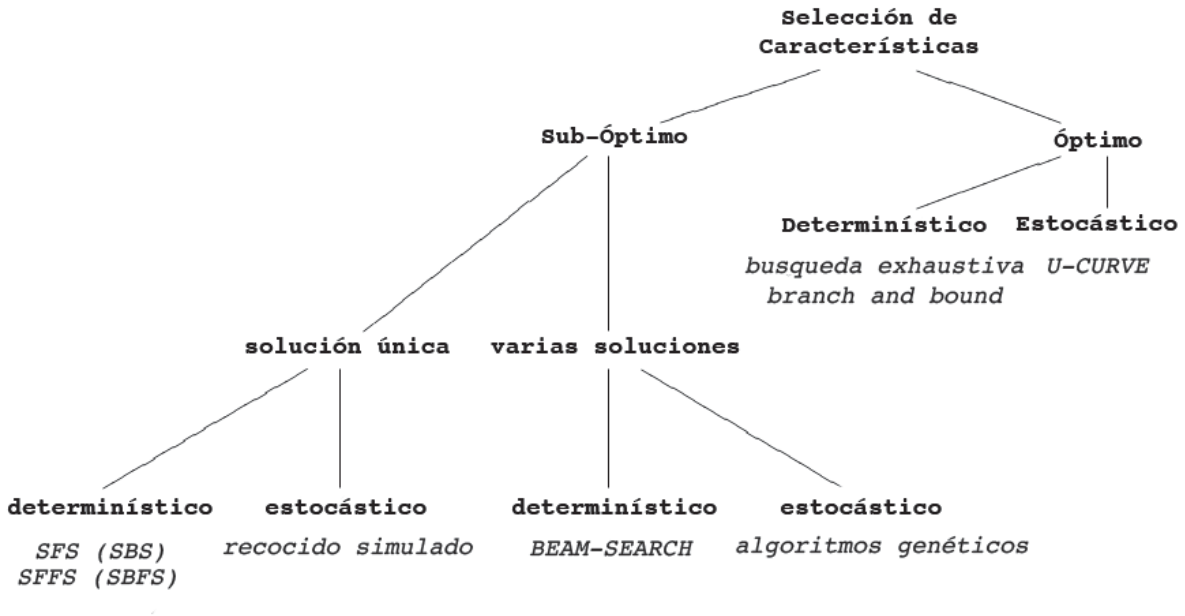


Figura 2.8: Categorización de los algoritmos de selección de características comúnmente empleados en el reconocimiento de patrones, figura adoptada de [Reis,].

Búsqueda Secuencial para el Frente y para Atrás

La búsqueda secuencial para el frente (del inglés: *Sequential Forward Search* - SFS) es un algoritmo de búsqueda golosa que comienza con un subconjunto resultado vacío y adiciona la mejor característica encontrada a este subconjunto. En seguida, adiciona una segunda característica que, en conjunto con la primera, forma el mejor par de características, y así sucesivamente. La Búsqueda Secuencial para Atrás (del inglés: *Sequential Backward Search* - SBS) es el algoritmo dual del SFS que comienza con el conjunto completo y elimina sucesivamente las características menos relevantes de acuerdo con la función criterio adoptada [Pudil et al., 1994]. A continuación mostraremos los algoritmos con los cuales realizaremos los experimentos para la evaluación del método de *Agrupamiento Lineal*.

Busqueda Exhaustiva Incremental (BEI)

La búsqueda exhaustiva incremental (BEI) se basa en la aplicación del algoritmo de búsqueda exhaustiva en busca del mejor subconjunto de características de un grado fijo k de

acuerdo con una función criterio adoptada. Inicialmente, este método obtiene la mejor característica individual (grado 1) y verifica que la mejor característica obtiene una mejora sobre un conjunto vacío de atributos (en este caso, se utiliza para la comparación el valor de la entropía *a priori* del gen objetivo considerado). Si hay una mejora, busca el mejor par de características (grado 2) comparando este subconjunto con la mejor característica individual obtenida (grado 1). El procedimiento termina cuando el mejor subconjunto de grado k' no mejora la función criterio con respecto al mejor subconjunto de grado $k'-1$, volviendo así el mejor subconjunto de grado $k'-1$. El algoritmo 1 presenta una descripción más precisa de este procedimiento.

Algorithm 1 Algoritmo de busqueda exhaustiva incremental (BEI)

Require: Conjunto de todos los genes $\mathbf{X}$, gen objetivo Y, conjunto de datos de expresión genética, y función criterio $\mathcal{F}$ que debe ser minimizada
Ensure: Conjunto de predictores $\mathbf{Z} = \{z_1, \ldots, z_k\}$ para Y
1: $\mathbf{Z} \leftarrow \emptyset$
2: $k \leftarrow 1$
3: $\mathbf{Z}' \leftarrow$ mejor subconjunto de tamaño 1 de $\mathbf{X}$ como predictor de Y de acuerdo con $\mathcal{F}$
4: **while** $\mathcal{F}(\mathbf{Z}', Y) < \mathcal{F}(\mathbf{Z}, Y)$ **do**
5: $\quad \mathbf{Z} \leftarrow \mathbf{Z}'$
6: $\quad k \leftarrow k + 1$
7: $\quad \mathbf{Z}' \leftarrow$ mejor subconjunto de tamaño k de $\mathbf{X}$ como predictor de Y
8: **end while**
9: **return** $\mathbf{Z}$

Busqueda Exhaustiva Secuencial para Atrás (SBS-E)

El algoritmo de Búsqueda Exhaustiva Secuencial para Atrás (SBS-E) primeramente aplica la búsqueda exhaustiva para el mejor subconjunto de tamaño k' inicial pré-definido. En seguida, se aplica la Búsqueda Secuencial para Atrás (SBS) a partir de ese conjunto conforme descrito en la Sección 2.7.3, el algoritmo termina cuando el mejor subconjunto de tamaño $k-1$ resulta ser peor que el tamaño del subconjunto k. El algoritmo 2 contiene una descripción mas precisa de este procedimiento.

2.7.4. Función Criterio

El problema central en reconocimiento de patrones es proyectar clasificadores a partir de un conjunto de evaluación, en este caso los datos son fortalecidos por una distribución

Algorithm 2 Algoritmo de búsqueda exhaustiva secuencial para atrás (SBS-E)

Require: Conjunto de todos los genes $\mathbf{X}$, gen objetivo Y, conjunto de datos de expresión genética, función criterio $\mathcal{F}$ a ser minimizada, y grado k' inicial
Ensure: Conjunto de predictores $\mathbf{Z} = \{z_1, \ldots, z_k\}$ para Y
1: $\mathbf{Z} \leftarrow$ mejor subconjunto de tamaño k' de $\mathbf{X}$ como predictor de Y
2: $\mathbf{Z}' \leftarrow \mathbf{Z} - \{Z^*\}$ tal que $Z^* = argmin_{Z \in \mathbf{Z}} \mathcal{F}(\mathbf{Z} - \{Z\}, Y)$
3: **while** $\mathcal{F}(\mathbf{Z}', Y) < \mathcal{F}(\mathbf{Z}, Y)$ e $|\mathbf{Z}| > 0$ **do**
4: $\quad \mathbf{Z} \leftarrow \mathbf{Z}'$
5: $\quad \mathbf{Z}' \leftarrow \mathbf{Z} - \{Z^*\}$ tal que $Z^* = argmin_{Z \in \mathbf{Z}} \mathcal{F}(\mathbf{Z} - \{Z\}, Y)$
6: **end while**
7: **return** $\mathbf{Z}$

conjunta de probabilidades de las configuraciones de un conjunto de características y de sus respectivos rótulos. Generalmente, tal distribución es estimada a partir de un número limitado de muestras, lo que justamente es un caso recurrente en inferencia de redes genéticas. El error de estimación depende de la selección de un conjunto de características que intenta predecir los rótulos a partir de los patrones observados. Una función criterio es una medida de cualidad de esta distribución conjunta estimada. La colección de la función criterio es fundamental, pues su papel es orientar los algoritmos de búsqueda por un subconjunto de características que mejor predice los rótulos a partir de las muestras con el objetivo de proyectar clasificadores que cometan pocos errores de rotulación (clasificación). Existen diversas funciones criterio propuestas en la literatura, dentro de las cuales se puede destacar:

- Distancia de Mahalanobis [Theodoridis and Koutroumbas, 2006]
- Distancia de divergencia [Duda et al., 2000]
- Distancia Kullback-Leibler [Theodoridis and Koutroumbas, 2006]
- Distancia de Bhattacharyya [Theodoridis and Koutroumbas, 2006]
- Coeficiente de determinación (CoD) [Dougherty et al., 2000]
- Entropía condicional media [Lin, 1991, Martins-Jr et al., 2006]

Estas funciones criterio estiman la distribución conjunta de un subconjunto de características, haciendo que sean susceptibles al fenómeno de la curva en U (Sección 2.7.1): para un número fijo de muestras, el aumento del número de características puede inducir a un au-

mento del error de estimación. De hecho, en la practica el número de muestras disponibles no es suficiente para realizar buenas estimaciones, requiriendo factores de penalización o maneras de reducir el número de parámetros estadísticos a ser estimados.

2.7.5. Entropía condicional media

La entropía condicional media esta siendo aplicada con satisfacción como función criterio para la selección de características en el contexto de la inferencia de redes genéticas [Martins Junior, 2009, Lopes et al., 2008, Lopes et al., 2009, Lopes et al., 2010, Martins-Jr et al., 2010, Lopes et al., 2011b]. La entropía mide el grado de desorden de una variable, o sea, cuán mayor es la entropía de una variable, mas difícil de predecir es su comportamiento. La entropía de una variable Y es dada por:

$$H(Y) = -\sum_{y \in Y} P(y) log P(y) \tag{2.11}$$

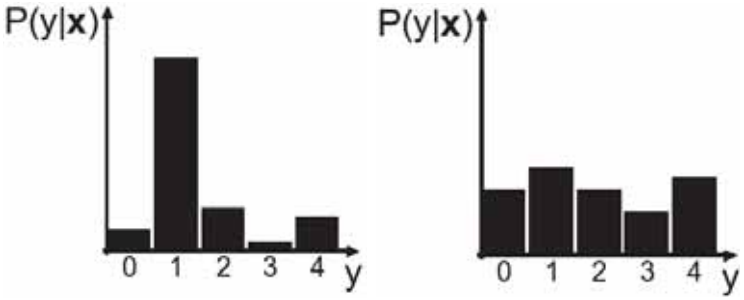
mientras que la entropía condicional de una variable Y dada una instancia $\mathbf{x} \in \mathbf{X}$ es definida por:

$$H(Y|\mathbf{x}) = -\sum_{y \in Y} P(y|\mathbf{x}) log P(y|\mathbf{x}) \tag{2.12}$$

donde $P(y|\mathbf{x})$ es la probabilidad condicional de $Y = y$ dado que $\mathbf{X} = \mathbf{x}$.

La entropía condicional dice en cuanto la variable $\mathbf{X}$ consigue predecir el comportamiento de la variable Y. Cuanto menor la entropía condicional de Y dado $\mathbf{X}$, mejor será la predicción de Y a través de $\mathbf{X}$. La Figura 2.9 ilustra dos histogramas, una para baja entropía condicional y otro para la alta entropía condicional de Y dado $\mathbf{x}$. El histograma de la izquierda (baja entropía condicional) configura el caso en que $\mathbf{x}$ realiza una buena predicción de los valores de Y, ya que la distribución de probabilidades condicionales presenta una masa altamente concentrada sobre $Y = 1$. Ya el histograma de la izquierda (alta entropía condicional) representa el caso en que $\mathbf{x}$ no es adecuado para predecir el comportamiento de Y, pues la distribución de probabilidades condicionales es próxima de

la uniforme.



$H(Y|\mathbf{x})$ baja $H(Y|\mathbf{x})$ alta

Figura 2.9: El histograma de la izquierda configura una situación en que Y es bien predicho por $\mathbf{x}$ porque la masa de probabilidades condicionales está bien concentrada en $Y = 1$ (entropía condicional baja). Ya para el histograma de la derecha, la masa de probabilidades está mejor distribuida a lo largo de las clases, lo que hace con que el patrón $\mathbf{x}$ no sea un buen predictor de Y (entropía condicional alta), figura adoptada de [Martins Junior, 2009].

La entropía condicional media es definida como la media ponderada de las entropías condicionales de todas las posibles instancias $\mathbf{x} \in \mathbf{X}$. Su ecuación es dada por:

$$H(Y|\mathbf{X}) = \sum_{\mathbf{x}\in\mathbf{X}} P(\mathbf{x})H(Y|\mathbf{x}) \tag{2.13}$$

en que $H(Y|\mathbf{x})$ es la entropía condicional dada por la Ecuación 2.12. Cuanto menor el valor de $H(Y|\mathbf{X})$, mayor será la ganancia de información sobre Y a través del conjunto de características $\mathbf{X}$.

Para procesar el error de estimación debido a los vectores de características de alta dimensión el número de muestras es insuficiente, una alternativa es penalizar las instancias no observadas en el cálculo de la función criterio y otra alternativa es no tomar en cuenta las instancias que no fueron observadas, a continuación mostramos a más detalle las penalizaciones mencionadas.

Penalización de las instancias no observadas

Para incorporar el error de estimación debido a los vectores de características de alta dimensión el número de muestras es insuficiente, una alternativa es aplicar las instancias no observadas en el cálculo de la función criterio [Martins Junior, 2009]. Una masa positiva de probabilidades es atribuida a las instancias no observadas, haciendo que el valor de la función criterio correspondiente a ellas se haga igual al valor de la función sin cualquier observación adicional (valor de la función para el conocimiento **a priori** de Y).

En el caso de la entropía condicional media (Ecuación 2.13), las instancias no observadas reciben la entropía de la distribución **a priori** de Y ($H(Y)$), La masa de probabilidades para las instancias no observadas es parametrizada por α. Este parámetro es adicionado a la frecuencia absoluta (número de observaciones) de todas las instancias posibles. Es así, que la entropía condicional media con este tipo de penalización es redefinida por:

$$H(Y|\mathbf{X}) = \frac{1}{\alpha M + s}\left[\alpha(M-N)H(Y) + \sum_{i=1}^{N}(f_i + \alpha)H(Y|\mathbf{X} = \mathbf{x_i})\right] \tag{2.14}$$

en que M es el número de instancias posibles del vector de características $\mathbf{X}$, N es el número de instancias observadas (así, el número de instancias no-observadas es dada por $M - N$), f_i es la frecuencia absoluta (número de observaciones) de $\mathbf{x_i}$ y s es el número de muestras del conjunto de entrenamiento. Las muestras $\mathbf{x_i}$ para $1 \leq i \leq N$ son aquellas observadas (así, las no observadas son $\mathbf{x_j}$ para $N + 1 \leq j \leq M$).

El cuadro 2.1 muestra un ejemplo conteniendo una tabla de frecuencias absolutas de los valores de (X_1, X_2, X_3, Y) y sus respectivas entropías condicionales y errores **a posteriori**, con y sin penalización de instancias no observadas, con $\alpha = 1$. La última fila de la tabla ("Total") muestra las respectivas entropías condicionales medias y los errores **a posteriori** medios.

La última fila de la tabla, denotada por "Total", indica las frecuencias absolutas totales y las medias ponderadas de las entropías condicionales y de los errores **a posteriori**.

Cuadro 2.1: Cuadro de frecuencias absolutas de (X_1, X_2, X_3, Y) y sus respectivas entropías condicionales y errores de predicción, con y sin penalización de instancias raramente observadas para $\alpha = 1$. Las frecuencias absolutas para $Y = 0$ e $Y = 1$ son indicadas respectivamente por $f(Y = 0)$ y $f(Y = 1)$. $f(\mathbf{x})$ y $f'(\mathbf{x})$ son las frecuencias absolutas de la instancia $\mathbf{X} = \mathbf{x}$ sin y con penalización, respectivamente. $H(Y|\mathbf{x})$ y $H'(Y|\mathbf{x})$ son las entropías condicionales sin y con penalización, respectivamente. La última fila de la tabla, denotada por "Total", indica las frecuencias absolutas totales e las medias ponderadas de las entropías condicionales y de los errores **a posteriori**. Para este ejemplo, $H(Y) = 0{,}9932$

X_1	X_2	X_3	$f(Y=0)$	$f(Y=1)$	$f(\mathbf{x})$	$f'(\mathbf{x})$	$H(Y\|\mathbf{x})$	$H'(Y\|\mathbf{x})$
0	0	0	7	3	10	11	0.8813	0.8813
0	0	1	1	0	1	2	0	0
0	1	0	0	0	0	1	0	0.9932
0	1	1	1	4	5	6	0.7219	0.7219
1	0	0	4	8	12	13	0.9183	0.9183
1	0	1	0	1	1	2	0	0
1	1	0	1	1	2	3	1	1
1	1	1	0	0	0	1	0	0.9932
Total			14	17	31	35	0.8207	0.8843

2.7.6. Coeficiente de Determinación

El coeficiente de determinación (del inglés: *Coefficient of Determination - CoD*) [Dougherty et al., 2007] es un criterio útil para la selección de características [Hsing et al., 2005]. Su ecuación viene dada por:

$$CoD_Y(\mathbf{X}) = \frac{\varepsilon_Y - \varepsilon_Y(\mathbf{X})}{\varepsilon_Y} \tag{2.15}$$

en que ε_Y es el error *a priori* y $\varepsilon_Y(\mathbf{X})$ es el error medio del clasificador óptimo (Bayesiano) a predecir Y tomando como entrada todas las instancias posibles de $\mathbf{X}$. El CoD cuantifica la disminución del error *a priori* en la observación de las instancias de $\mathbf{X}$. Cuanto mayor sea el CoD, mayor será esta disminución ($CoD = 0$ significa que $\mathbf{X}$ no mejora el desempeño del clasificador óptimo en relación al error *a priori*, en cuanto $CoD = 1$ indica que Y está completamente determinado por $\mathbf{X}$).

2.8. Ciclo de vida de la Malaria

La malaria es una enfermedad parasitaria transmitida por la picadura de los mosquitos hembras del género *Anopheles* infectados con protozoos del género *Plasmodium* y se caracteriza por episodios de fiebre con ciertos intervalos, dependiendo de la especie de Plasmodium implicado en la naturaleza. Existen cuatro especies que causan enfermedades en los seres humanos: *Plasmodium falciparum*, *Plasmodium Vivax*,*Plasmodium Malariae* y el *Plasmodium Ovale*. En el Perú, se encuentran sólo dos especies de *Plasmodium*: *Plasmodium Vivax* y *Plasmodium falciparum*, la forma mas grave es producida por el *Plasmodium falciparum* (terciana maligna), para mas información [de Salud Dirección General de Epidemiología, 2014]

El ciclo de vida del *Plasmodium* comienza cuando el mosquito ingiere sangre de un huésped humano que contienen formas maduras sexuales, llamadas gametocitos masculinos y femeninos. En el estómago del mosquito los gametocitos masculinos sufren exflagelación, originando gametas masculinos (microgametas), mientras que los gametocitos femeninos sufren maduración para gametas femeninos (macrogametas). La microgameta, entonces fertiliza la macrogameta, y la fusión de estos dos forman el cigoto, que luego de convertirse en movible se llama oocineto. El oocineto se fija en la pared del estómago, entre las células epiteliales, siendo nombrado ooquistes. Dentro de los ooquistes se forman muchos esporozoítas, celulas alargadas y móviles, con un núcleo central, que después de la liberación en la cavidad del cuerpo del mosquito, alcanzan a las glándulas salivales haciéndola infectivo. Cuando el mosquito se alimenta de sangre otra vez, las esporozoítas se inyecta en la corriente circulatoria. Después de un máximo de 30 minutos, las esporozoítas entran en los hepatocitos y sufren el proceso de desarrollo conocido como esquizogonia pré-eritrocítica. Es interesante señalar que las esporozoítas puede entrar y salir de muchos hepatocitos, a través de la penetración directa en la membrana plasmática, probablemente buscando los hepatocitos que son susceptibles a la infección específica [M. et al., 2001]. Los esquizontes que se forman después de la esquizogonia pré-eritrocítica rompen la célula y liberan merozoítos en la circulación, desarrollando a su vez la invasión de los eritrocitos (glóbulos

rojos) y convertiendolos en trofozoítos jovenes o anillos. Cuando empiezan a crecer y su citoplasma se vuelve más irregular, se conocen como maduros, y después el proceso de división asexual, son llamados de esquizontes que contiene varios merozoítos en su interior. Al finalizar el proceso de división los esquizontes rompen los eritrocitos y los merozoitos liberados en la corriente circulatoria invaden nuevos eritrocitos, iniciando de nuevo el ciclo eritrocítico. Esta liberación de parásitos en la circulación es responsable de la aparición de síntomas y por lo tanto la duración del ciclo eritrocítico determina la frecuencia de estos síntomas, variando entre las especies del *Plasmodium*. Después de algunas generaciones de merozoitos, algunas se diferencian en formas sexuales (gametocitos), sufren maduración son ingeridas por las hembras de anofelinos durante la comida sanguinea, cerrando el ciclo. (Figura 2.10)

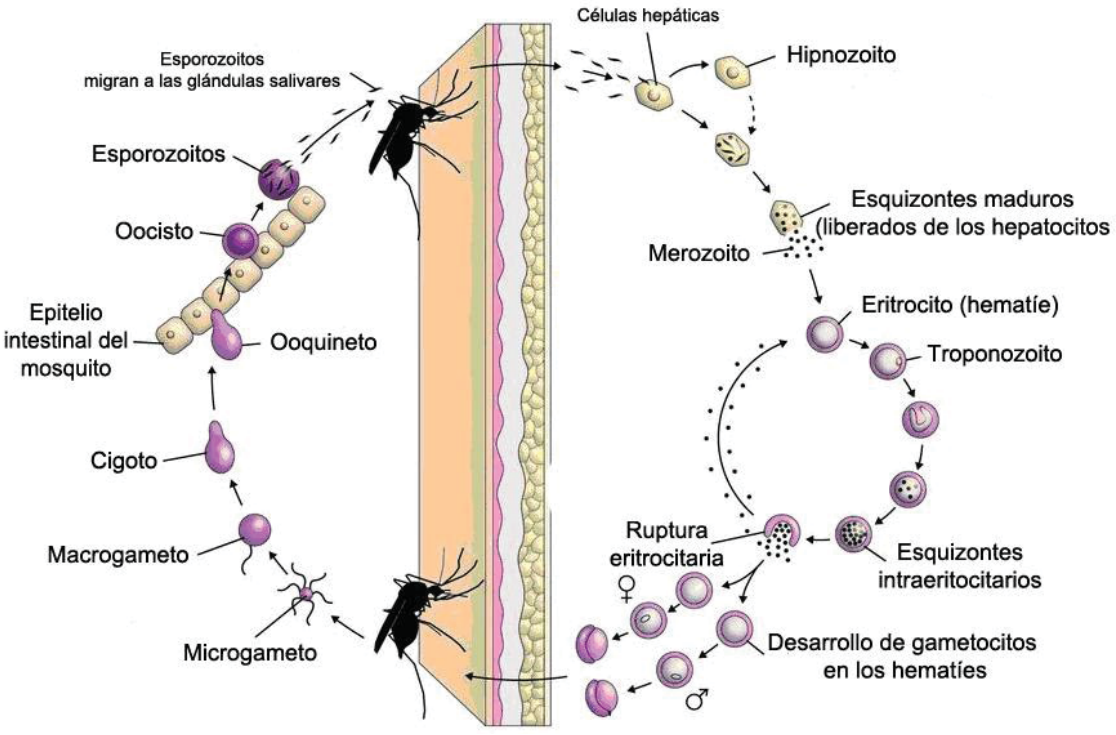


Figura 2.10: Ciclo de vida de la Malaria, figura adoptada de [Hopkins, 2015].

Capítulo 3

Inferencia en Redes Genéticas

La combinación del análisis de la expresión, disturbio, tratamientos y las mutaciones de los genes pueden indicar informaciones de efectos moleculares o funciones especificas de los genes. Inferir redes reguladoras genéticas de datos de expresión, también conocida como proceso de ingeniería inversa, no es una tarea computacional fácil debido al enorme volumen de datos (genes) y al pequeño número de muestras (medidas) disponibles, incluyendo redes biológicas altamente complejas, que representa un importante objetivo en la investigación de la bioinformática y la biología computacional [Hovatta, 2005]. La inferencia de GRNs a partir de datos temporales de expresión busca identificar la variación de los niveles de expresión a lo largo del tiempo, haciendo posible indicar información tal como: diferentes vías de regulación, ciclo celular y mapeo de cambios causados por estímulos, sirviendo como un modelo para la representación funcional de las interacciones genéticas. Es importante destacar que la inferencia de GRNs pretende encontrar redes de interacción entre los genes que son potencialmente interesantes bajo el punto de vista biológico de la observación de sus expresiones. De esta manera, ella sugiere la relación entre los genes según un estimador para limitar el número de interacciones. Como estos experimentos tienen un alto costo financiero, humano y de tiempo, la idea es ofrecer al biólogo experto una visualización de un conjunto reducido de interconexiones entre los genes, en la forma de un GRN, en el cual tendrá la oportunidad de generar hipótesis de manera integrada sobre un determinado fenómeno de interés [Martins Junior, 2009].

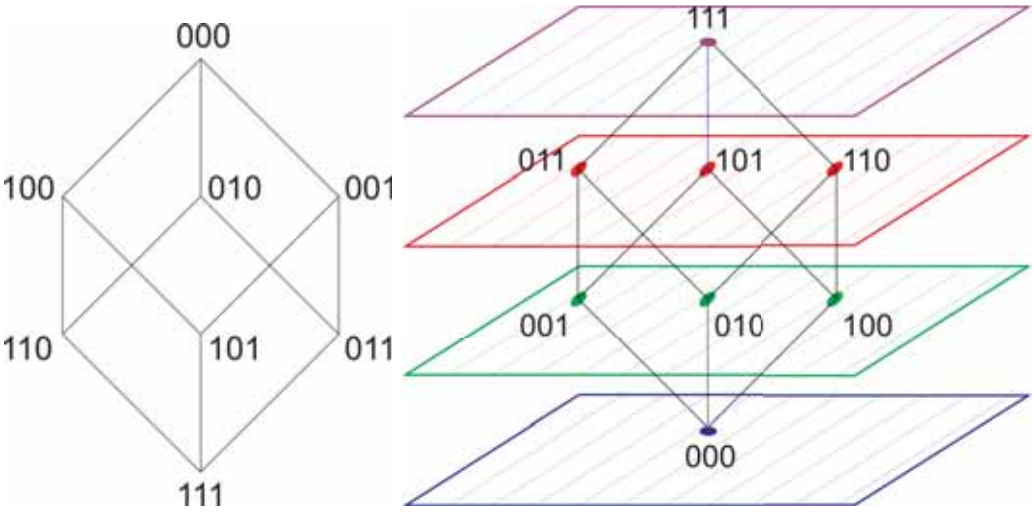
3.1. Selección de Características por Agrupamiento Lineal

En el marco probabilístico adoptado, es necesario estimar las distribuciones de las probabilidades condicionales de los valores del gen candidato dado las configuraciones posibles de los valores predictores. El pequeño número de muestras disponibles dificulta la estimación de esas probabilidades, especialmente al analizar conjuntos con mayor numero de predictores. Si el número de muestras es mantenido fijo y la cantidad de predictores fuera suficientemente grande, entonces parte de las configuraciones de las variables predictores no serán observadas en los datos experimentales. Y aún considerando las configuraciones observadas, el número de ocurrencias de buena parte de sus configuraciones puede ser insuficiente para una estimación confiable de la distribución de probabilidades condicionales de los valores del gen objetivo dadas estas configuraciones. Esto ejemplifica el fenómeno de la *maldición de la dimensionalidad* (ver sección 2.7.1).

Existen métodos en la literatura para lidiar con el problema de la dimensionalidad, tales como la entropía condicional media con penalización de instancias no observas (ecuación 2.14). Aún así, a medida que el numero de predictores aumenta, las configuraciones observadas resultan tan enrarecidas que el poder de estimación de estos métodos acaba quedando cada vez mas comprometido. En un intento de minimizar este problema, en este proyecto son utilizadas algunas técnicas para reducir el número de parámetros a ser estimados, agrupando configuraciones interrelacionadas en grupos que representan (clases de equivalencia).

Siguiendo el objetivo propuesto, se adopta el modelo de Redes Booleanas para modelar los genes y sus interacciones, los cuales serán explotados particionamientos de reticulados booleanos por medio de hiperplanos con el objetivo de reducir el número de parámetros a ser estimados, de forma que minimice el problema de dimensionalidad. Un ejemplo de este tipo de particionamiento es mostrado en la figura 3.1(b). Así, al realizar los cortes por hiperplanos en el reticulado, las configuraciones que están en la misma partición pertenecen a la misma clase de equivalencia, como se ve en la figura 3.1(b). Una de las maneras de hacer esto es a través de una agrupación por combinaciones lineales (por lo tanto, de

aquí en adelante este enfoque será llamado agrupamiento lineal), que consigue reducir el número de configuraciones de 2^k para $k+1$ configuraciones, siendo k la cardinalidad (o grado) del conjunto de predictores considerados.



(a) (b)

Figura 3.1: (a) Reticulado Booleano de tres genes. (b) Cortes de hiperplanos sobre un reticulado de tres genes, generando cuatro particiones: $\{000\}$; $\{100, 010, 001\}$; $\{110, 101, 011\}$; $\{111\}$, figura adoptada de [Montoya, 2014].

3.1.1. Inferencia en Redes Genéticas Probabilísticas

Una red genética probabilística (PGN) es representada por un grafo $\mathcal{G} = (\mathbf{X}, \Phi)$ con n nodos $\mathbf{X} = \{X_1, X_2, ...X_n\}$, $X_i \in \mathcal{D}$, y un vector de n funciones $\Phi = (\phi_1, \phi_2, ..., \phi_n)$, $\phi_i : \mathcal{D}^{k_i} \rightarrow \mathcal{D}$ donde k_i es el número de genes de los cuales el gen X_i depende (grado de entrada de X_i). En el caso particular de redes booleanas, $\mathcal{D} = \{0, 1\}$. En este enfoque, se considera que cada gen $Y \in \mathbf{X}$ está asociado a un subconjunto de genes predictores $\mathbf{Z} = \{Z_1, Z_2, ..., Z_k\} \subseteq \mathbf{X}$ y que el valor $Y[t+1]$ del gen en un instante $t+1$ es determinado únicamente por la configuración $\mathbf{Z}[t]$ de sus predictores en el instante anterior t. Por lo tanto, para cada gen, se puede aplicar un algoritmo de selección de características para verificar cual subconjunto de genes predice mejor el comportamiento del gen considerado. Una vez determinado el mejor subconjunto, este estará ligado al gen considerado en la red resultante. Una ilustración de como esta estimación es realizada para un único gen objetivo puede ser verificada en la figura 3.2. Esta consiste en una matriz de expresiones temporales

binarias en la cual cada linea representa un perfil de expresión de un determinado gen y las columnas representan instantes de tiempo (en este ejemplo, de 1 a 48). El procedimiento representado por esta figura se inicia con el llenado de la tabla de las frecuencias absolutas de Y en el instante de tiempo posterior $(t+1)$ dadas cada una de las instancias de (X_1, X_2) en el instante de tiempo actual (t), para $1 \leq t \leq m-1$, donde m es el número de muestras temporales. En seguida, esta tabla de cuentas es transformada en una tabla de distribución conjunta de probabilidades, sobre la cual una función criterio (ejemplo: entropía condicional media) puede ser aplicada. Tal procedimiento es repetido para todos los subconjuntos examinados por el algoritmo de selección de características adoptado (ejemplo: en el caso del algoritmo de búsqueda exhaustiva con un grado fijo, en este ejemplo todos los pares de predictores serán examinados). El mejor subconjunto clasificado de acuerdo con la función criterio adoptada es anexado como predictor del gen Y en la red.

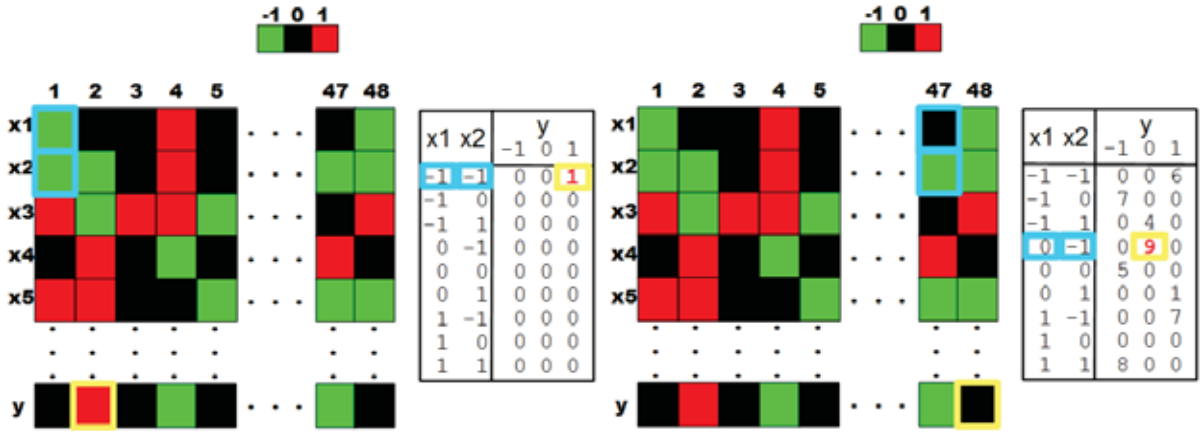


Figura 3.2: Procedimiento de obtención de las frecuencias absolutas del gen objetivo Y dadas las instancias de los candidatos a los predictores (X_1, X_2) a partir de una matriz de expresiones temporales binarias compuesta por 48 instancias de tiempo en que los genes $X_1, X_2, ..., X_n$ son dispuestos en lineas y los tiempos $(t = 1, ..., 48)$ en columnas. La figura de la izquierda muestra el inicio del procedimiento (recolección de la primera muestra $(x_1[1], x_2[1], y[2])$) y la de la derecha muestra el final del procedimiento (recolección de la ultima muestra $x_1[47], x_2[47], y[48]$). Los colores verde, negro y rojo representan, respectivamente, los valores -1 (subexpresion), 0 (normal) y 1 (superexpresion), figura adoptada de [Martins Junior, 2009].

La entropía condicional media, descrita en la Sección 2.7.5, es tomada como función criterio para la construcción de las PGNs. La información mutua [Butte and Kohane, 2000] depende de la entropía en la distribución de probabilidades *a priori* del gen objetivo ($H(Y)$) y de la entropía de la distribución condicional del objetivo dado los predictores ($H(Y|\mathbf{Z})$), $\mathbf{Z} \subseteq \mathbf{X}$. Como el orden de clasificación de los subconjuntos no dependen de $H(Y)$, ya que $H(Y)$ es siempre el mismo para cualquier subconjunto considerado, esta ordenación depende apenas de las entropías condicionales. El método de estimación de la entropía condicional a ser adoptado en este trabajo esta basado en la penalización de las instancias no observadas (ver Sección 2.7.5, Ecuación 2.14).

3.2. Agrupamiento Lineal

Para reducir el número de parámetros necesarios para la estimación de las distribuciones de probabilidades condicionales $P(Y|\mathbf{Z})$, donde Y es una variable binaria y $\mathbf{Z} \subseteq \mathbf{X}$ es un vector binario en $\{0,1\}^k$, utilizaremos el método denominado *Agrupamiento Lineal*. Tal método realiza una reducción del número de configuraciones a ser consideradas para la estimación, de 2^k para $k+1$, a través de un mapeo lineal del vector de entrada $\mathbf{Z} \in \{0,1\}^k$ en un número entero L. El mapeo es definido por una combinación lineal, expresado por la ecuación 3.1:

$$L = a_1 Z_1 + a_2 Z_2 + \ldots + a_k Z_k \tag{3.1}$$

en que $a_i \in \{-1, 1\}$, para $i \in \{1, 2, \ldots, k\}$. Podemos definir un vector de coeficientes $\mathbf{A} = \{a_1, a_2, ..., a_k\} \in \{-1, +1\}^k$, en seguida reescribiendo la ecuación 3.1 en formato vectorial: $L = \mathbf{A}^T\mathbf{Z}$. Este enfoque, supone que un predictor pueda ser un activador (caso su coeficiente sea $+1$) o un inhibidor (caso su coeficiente sea -1) del gen objetivo. Como cada coeficiente a_i puede asumir dos valores, existen 2^k combinaciones lineales posibles a ser evaluadas para cada conjunto de k predictores. Para un determinado conjunto de predictores, se adopta la configuración de coeficientes $\mathbf{A}^*$ que resulta el mejor valor de la función criterio. La comparación entre diferentes conjuntos predictores será basada en el valor de la función criterio para la mejor combinación lineal de cada conjunto.

Como los predictores Z_i son variables aleatorias discretas (binarias) y los coeficientes a_i

son números enteros, L es una variable aleatoria discreta también. El número de valores posibles para L es exactamente $k+1$ para cualquier configuración del vector de coeficientes $\mathbf{A}$. La demostración de este hecho es como sigue. Sea k^- y k^+ el número de coeficientes negativos (inhibidores) y el número de coeficientes positivos (activadores) respectivamente, siendo entonces $k = k^- + k^+$. Como las expresiones genéticas son valores binarios (0 ó 1), el menor valor que L puede asumir es exactamente $-k^-$, que acontece cuando todos los genes inhibidores están conectados (1) y los genes activadores están desconectados (0). Análogamente, el mayor valor que L puede asumir es k^+, ocurriendo cuando todos los genes inhibidores están desconectados (0) y los genes activadores están conectados (1). Por tanto $L \in \{-k^-, -k^- + 1, ..., k^+ - 1, k^+\}$, siendo la cardinalidad de este conjunto dado por $k^+ - (-k^-) + 1 = k + 1$, como fue demostrado.

El resultado encima implica que las 2^k configuraciones de el vector original de predictores son mapeados para $k + 1$ clases de equivalencia, de acuerdo con los valores resultantes de una determinada combinación lineal de los valores de los predictores. En el método utilizado, la estimación directa de $P(Y|\mathbf{Z})$ es sustituida por la estimación de $P(Y|L)$. De este modo el número de parámetros a ser estimados a partir del conjunto de muestras resulta bastante reducido. En seguida, se puede aplicar cualquier función criterio para la evaluación de los predictores, por ejemplo, la entropía condicional media penalizada (Ecuación 2.14), utilizándose $P(Y|L)$ en lugar de $P(Y|\mathbf{Z})$.

A pesar de existir 2^k diferentes formas de definir el vector de coeficientes $\mathbf{A}$ como dicho anteriormente, apenas la mitad de estas formas (2^{k-1}) precisan ser evaluadas, pues la otra mitad genera exactamente los mismos agrupamientos con la señal de L invertido. De hecho, si $\mathbf{A}' = -\mathbf{A}$, entonces $\mathbf{A}'$ y $\mathbf{A}$ generan los mismos grupos donde las magnitudes de sus respectivas combinaciones lineales son las mismas, más con las señales invertidas. La Tabla 3.1 muestra un ejemplo de esto.

Para ilustrar el método, consideremos la tarea de estimar el valor de $Y \in \{0, 1\}$ a partir de las variables $Z_1, Z_2, Z_3 \in \{0, 1\}$. Esto lleva a $2^3 = 8$ configuraciones distintas, lo que implica la necesidad de estimar $P(y|z_1, z_2, z_3)$ para 8 configuraciones de z_1, z_2, z_3 y dos valores de y. La Tabla 3.2 muestra el cálculo de la entropía condicional media sobre estas 8 configuraciones originales (sin agrupamiento).

Cuadro 3.1: Agrupamiento de configuraciones en particiones con $(a_1, a_2, a_3) = (-1, -1, -1)$ e $(a_1, a_2, a_3) = (+1, +1, +1)$. Note que ambos agrupamientos generan los mismos grupos.

$(a_1, a_2, a_3) = (-1, -1, -1)$		$(a_1, a_2, a_3) = (+1, +1, +1)$	
$L_1 =$	$-1z_1 - 1z_2 - 1z_3$	$L_2 =$	$+1z_1 + 1z_2 + 1z_3$
-3	111	3	111
-2	011,101,110	2	011,101,110
-1	001,010,100	1	001,010,100
0	000	0	000

Cuadro 3.2: Cálculo de la entropía condicional media con penalización de instancias no observadas de acuerdo con la ecuación 2.14. Note que la configuración $(z_1, z_2, z_3) = (1, 0, 1)$ no fue observada, sufriendo así una penalidad al obtener el valor de la entropía condicional igual al valor de la entropía *a priori* de Y ($H(Y) = 0,9641$ en este caso).

z_1	z_2	z_3	$f(Y=0)$	$f(Y=1)$	$H(Y\|z_1, z_2, z_3)$
0	0	0	2	3	0.9710
0	0	1	1	1	1.0
0	1	0	1	0	0.0
0	1	1	0	1	0.0
1	0	0	2	1	0.9183
1	0	1	0	0	0.9641
1	1	0	0	1	0.0
1	1	1	1	4	0.7219
Total			7	11	$H(Y\|\mathbf{Z}) = 0,7215$

A pesar de existir $2^3 = 8$ combinaciones lineales posibles para el ejemplo de la Tabla 3.2, apenas 4 diferentes combinaciones lineales precisan ser investigadas: $(-1, -1, -1)$, $(-1, -1, +1)$, $(-1, +1, -1)$, $(-1, +1, +1)$. Las otras 4 combinaciones restantes generan exactamente las mismas particiones como ya discutido anteriormente. La figura 3.3 representa los particionamientos en el reticulado para cada una de estas combinaciones lineales, mientras que las tablas 3.3, 3.4, 3.5 y 3.6 muestran los cálculos de las entropías condicionales medias para estas combinaciones lineales considerando el mismo objetivo y los mismos predictores de la tabla 3.2.

Así, dentro de los 4 valores de la entropía condicional media obtenidos, uno para cada combinación lineal, es escogido aquel con el mejor valor para ser el representante del

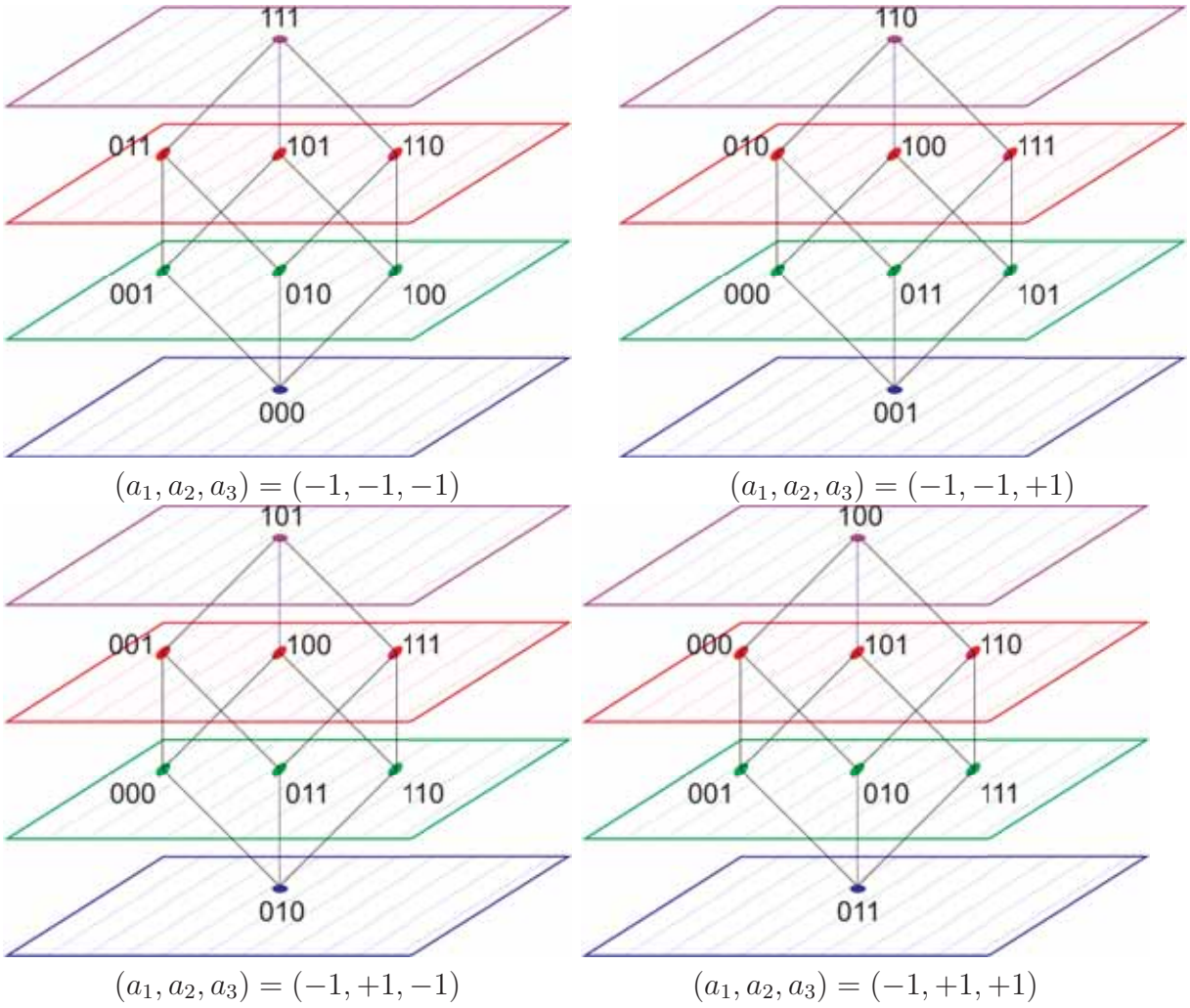


Figura 3.3: Particionamientos en el reticulado Booleano para los coeficientes lineales $(a_1, a_2, a_3) = \{(-1, -1, -1); (-1, -1, +1); (-1, +1, -1); (-1, +1, +1)\}$, figura adoptada de [Montoya, 2014]

Cuadro 3.3: Aplicación del agrupamiento lineal $(a_1, a_2, a_3) = (-1, -1, -1)$ sobre el ejemplo de la tabla 3.2

$L =$	$-1z_1 - 1z_2 - 1z_3$	$f(Y = 0)$	$f(Y = 1)$	$H(Y\|z_1, z_2, z_3)$
-3	111	1	4	0.7220
-2	011,101,110	0	2	0.0
-1	001,010,100	4	2	0.9183
0	000	2	3	0.9710
Total		7	11	$H(Y\|\mathbf{Z}) = 0{,}7539$

conjunto predictor considerado, en este caso $H(Y|\mathbf{Z}) = 0{,}7539$, que es el valor de la Tabla 3.3 con el vector $\mathbf{A} = (-1, -1, -1)$. El Algoritmo 3 representa este proceso.

Cuadro 3.4: Aplicación del agrupamiento lineal $(a_1, a_2, a_3) = (-1, -1, +1)$ sobre el ejemplo de la Tabla 3.2.

$L =$	$-1z_1 - 1z_2 + 1z_3$	$f(Y = 0)$	$f(Y = 1)$	$H(Y\|z_1, z_2, z_3)$
-2	110	0	1	0.0
-1	010,100,111	4	5	0.9911
0	000,011,101	2	4	0.9183
1	001	1	1	1.0
Total		7	11	$H(Y\|\mathbf{Z}) = 0{,}8790$

Cuadro 3.5: Aplicación del agrupamiento lineal $(a_1, a_2, a_3) = (-1, +1, -1)$ sobre el ejemplo de la Tabla 3.2.

$L =$	$-1z_1 + 1z_2 - 1z_3$	$f(Y = 0)$	$f(Y = 1)$	$H(Y\|z_1, z_2, z_3)$
-2	101	0	0	0.9641
-1	001,100,111	4	6	0.9710
0	000,011,110	2	5	0.8631
1	010	1	0	0.0
Total		7	11	$H(Y\|\mathbf{Z}) = 0{,}8870$

Cuadro 3.6: Aplicación del agrupamiento lineal $(a_1, a_2, a_3) = (-1, +1, +1)$ sobre el ejemplo de la Tabla 3.2.

$L =$	$-1z_1 + 1z_2 + 1z_3$	$f(Y = 0)$	$f(Y = 1)$	$H(Y\|z_1, z_2, z_3)$
-1	100	2	1	0.9183
0	000,101,110	2	4	0.9183
1	001,010,111	3	5	0.9544
2	011	0	1	0.0
Total		7	11	$H(Y\|\mathbf{Z}) = 0{,}8496$

El agrupamiento de las configuraciones naturalmente implica una perdida de información, ya que este proceso es análogo a un proceso de cuantización: la información sobre las configuraciones originales acaba siendo perdida. Esta perdida de información puede traer beneficios en el poder de la estimación en los casos en que las muestras no son suficientes para rellenar la tabla de frecuencias de manera satisfactoria (el número de muestras requerido crece exponencialmente en relación a la dimensión del conjunto de predictores). En los casos en que la tabla sea rellenada de manera satisfactoria, la aplicación del agrupamiento puede acabar siendo perjudicial. Es así que serán mencionadas dos variantes del método de agrupamiento lineal de modo que se aplique el agrupamiento cuando sea solo

Algorithm 3 Entropía Condicional Media por Agrupamiento Lineal

Require: Conjunto de predictores candidatos $\mathbf{Z} = \{z_1, \ldots, z_k\}$, gen candidato Y, matriz de expresión genética con m muestras temporales y n genes, y los 2^{k-1} posibles agrupamientos.

Ensure: Entropía condicional media de $\mathbf{Z} = \{z_1, \ldots, z_k\}$ dado Y.

1: $ECM_{min} \leftarrow \infty$
2: **for** cada posible agrupamiento definido por una instancia del vector de coeficientes lineales **A do**
3: Rellenar la tabla TPC de probabilidades condicionales de acuerdo con la matriz de expresión genética, los valores de $Y[t+1]$, y los valores resultantes de las combinaciones lineales de los valores de $\mathbf{Z}[t]$ para todo $1 \leq t \leq m-1$, y de los coeficientes lineales.
4: Calcular la entropía condicional media ECM con base en la tabla TPC
5: **if** $ECM < ECM_{min}$ **then**
6: $ECM_{min} \leftarrow ECM$
7: **end if**
8: **end for**
9: **return** ECM_{min}

necesario.

Algorithm 4 Entropía Condicional Media por Agrupamiento Lineal Selectivo

Require: Conjunto de predictores candidatos $\mathbf{Z} = \{z_1, \ldots, z_k\}$, gen candidato Y, matriz de expresión genética con m muestras temporales y n genes, y los 2^{k-1} posibles agrupamientos.

Ensure: Entropía condicional media de $\mathbf{Z} = \{z_1, \ldots, z_n\}$ dado Y.

1: **if** todas las instancias de $\mathbf{Z}$ debieron haber sido observadas por lo menos una vez en la matriz de expresión genética **then**
2: Rellenar la tabla TPC de probabilidades condicionales de acuerdo con la matriz de expresión genética, los valores de $Y[t+1]$, y los valores de $\mathbf{Z}[t]$ para todo $1 \leq t \leq m-1$.
3: Calcular la entropía condicional media ECM con base en la tabla TPC
4: **return** ECM
5: **else**
6: **return** resultado del Algoritmo 3 (Agrupamiento Lineal)
7: **end if**

3.2.1. Agrupamiento Lineal Selectivo

En esta estrategia, el agrupamiento lineal solo es aplicado si por lo menos una de las instancias del conjunto de predictores no hubiera sido observada en las muestras (estos casos resultan en lineas de la tabla de frecuencias con valores nulos). Así, si todas las instancias poseerán por lo menos una observación, se toma la entropía condicional media

obtenida a partir de la tabla original sin agrupamiento. El Algoritmo 4 ilustra este proceso.

3.2.2. Agrupamiento Lineal Mixto

En esta estrategia, el valor de la entropía condicional media obtenida por la aplicación del agrupamiento lineal (Algoritmo 3) es comparada con el valor de la entropía condicional media sobre la tabla original, haciendo que el menor de ellos sea atribuido al conjunto de predictores considerado. El Algoritmo 5 describe este proceso.

Algorithm 5 Entropía Condicional Media por Agrupamiento Lineal Mixto

Require: Conjunto de predictores candidatos $\mathbf{Z} = \{z_1, \ldots, z_k\}$, gen candidato Y, matriz de expresión genética con m muestras temporales y n genes, y los 2^{k-1} posibles agrupamientos.

Ensure: Entropía condicional media de $\mathbf{Z} = \{z_1, \ldots, z_k\}$ dado Y.

1: Rellenar la tabla TPC de probabilidades condicionales de acuerdo con la matriz de expresión genética, los valores de $Y[t+1]$, y los valores de $\mathbf{Z}[t]$ para todo $1 \leq t \leq m-1$.

2: Calcular la entropía condicional media ECM con base en la tabla TPC
3: $ECM_{AL} \leftarrow$ resultado del Algoritmo 3
4: **if** $ECM_{AL} < ECM$ **then**
5: **return** ECM_{AL}
6: **else**
7: **return** ECM
8: **end if**

Capítulo 4

Resultados experimentales en datos del Plasmodium falciparum

4.1. Consideraciones preliminares

Para evaluar el método de *Agrupamiento lineal* en datos reales, fueron realizados los experimentos con datos de expresión genética del transcriptoma del ciclo de desarrollo intraeritrocitario (del inglés: *Intraerythrocytic Developmental Cycle* - IDC) del *Plasmodium falciparum*.
En la sección 4.1.1 muestra la obtención de los datos de expresión genética, los cuales sufren un proceso obligatorio de discretización para la estimación de la PGN (ver sección 2.4.2). A la vez los algoritmos de búsqueda y la función criterio son descritas en la sección 4.1.2 y 4.1.3 respectivamente.
Finalmente, los resultados experimentales muestran la red glicolítica al aplicar el método de agrupamiento lineal.

Todos los experimentos fueron ejecutados en los laboratorios de visión computacional del Instituto de Matemática y estadística (IME) de la Universidad de Sao Paulo (USP). El computador usado tiene las siguientes características:

- Procesador: Intel Xeon(R) CPU E5-2620 v2 @ 2.10GHz x 24 cores.

- Memoria: 156 GB
- Disco Duro: 10 TB
- Vídeo: Quadro K620/PCIe/SSE2

4.1.1. Datos del HB3

El genoma transcriptoma del ciclo de desarrollo intraeritrocítico (del inglés: *Intraerythrocytic Developmental Cycle* - IDC) del *Plasmodium falciparum* fue generado por medidas relativas de los niveles de abundancia del mARN (medidas transcripcionales) de muestras recogidas de una cepa llamada HB3, que es bien caracterizada y localizada en Honduras [Bozdech and DeRisi, 2005]. Del *microarray* original, representando este transcriptoma se construyó un conjunto de datos, llamado USP-dataset, resultado de la aplicación de un filtro de proceso previo que elimino los *spots* con señal muy débil en ambos canales Cy3 y Cy5 según un determinado umbral de corte [Barrera et al., 2004]. Como resultado de este tratamiento previo, USP-dataset contiene 5080 genes.

4.1.2. Algoritmo de búsqueda adoptada

Para encontrar los posibles predictores de la red se adopto el algoritmo de búsqueda exhaustiva incremental por el mejor par de predictores, dicho algoritmo es descrito en la Sección: 2.7.3.

4.1.3. Función criterio adoptada

La función criterio adoptada para evaluar la calidad de la predicción en relación al comportamiento de un determinado gen objetivo está basado en la entropía condicional media conforme a la ecuación 2.14. Fue adoptada $\alpha = 1$ como parámetro de penalización de las instancias no observadas. El método de agrupamiento lineal evaluado para tratar de mejorar la estimación de la entropía media condicional fueron presentados (ver Sección 2.7.4).

4.2. Experimentos

Se adoptaron los datos de la expresión genética del transcriptoma del ciclo de desarrollo intraeritrocitario (del inglés: *Intraerythrocytic Developmental Cycle* - IDC) del *Plasmodium falciparum* (un agente de la malaria) para evaluar el método de *agrupamiento lineal*. Tal transcriptoma se generó mediante mediciones relativas de los niveles de abundancia del mARN a partir de muestras recogidas de una cepa llamada HB3 (ver Sección: 4.1.1) [Bozdech and DeRisi, 2005]. El conjunto de datos de control de calidad (QC llamado conjunto de datos) que contiene 48 muestras con 5080 genes fueron utilizados en este experimento. Estás 48 muestras se extrajeron cada hora, correspondientes a las 48 horas (instantes de tiempo) del IDC. Dado que los datos son números reales, se aplicó un proceso de cuantificación en 3 niveles (sub expresión, expresión normal y súper expresión) (ver Sección: 2.4.2).

Barrera en [Barrera et al., 2007, Barrera et al., 2004] aplica exactamente el mismo método denominado aquí como "normal"(sin agrupar), tal como se describe anteriormente, para analizar la red glicolítica del parásito. Ellos adoptaron 10 genes conocidos para modificar enzimas que pertenecen a la vía de la glicólisis, estás son utilizadas como semillas para poner a prueba la capacidad de inferencia del modelo PGN propuesto por ellos. Una red fue generada mediante la selección de los 10 mejores predictores individuales para cada objetivo considerado.

Datos de entrada

El conjunto de datos utilizados en esta tesis fueron discretizados y normalizados en tres niveles de expresión $R = \{0, 1, 2\}$, de esta forma en los experimentos realizados tratamos de buscar los 10 mejores pares predictores para los genes objetivos de la glicólisis. En esta tesis se muestran 2 predictores por gen objetivo a diferencia de un predictor por gen mostrado en el resultado del libro [Barrera et al., 2007], de esta manera nosotros evaluamos el método con una tabla de frecuencia mas grande, llegando a evaluar la efectividad del método.

El conjunto de muestras utilizado en esta tesis consta de 48 muestras en total, nosotros mostramos solo la muestra # 1, las demás muestras están en el CD. Esta muestra contiene

el comportamiento de los 5048 genes, es por eso que podemos solo notar los valores $R = \{0, 1, 2\}$, estos valores nos indican si el gen se comporta como subexpreso (en el caso 0), como normal (en el caso 1) o como superexpreso (en el caso 2), esta muestra # 1 se puede observar en el anexo A.

Procesamiento

Para todos los experimentos realizados utilizamos java como plataforma de desarrollo, como resultado de tales experimentos nosotros tuvimos que adaptar el software utilizado en el trabajo de [Montoya, 2014], las clases principales se muestran en el anexo B.

Resultado

Los datos obtenidos luego de inferir las redes genéticas probabilísticas, se muestran en las tablas siguientes, nosotros mostramos solo los 10 mejores pares de predictores por gen objetivo, así como la información mutua calculada para cada gen, estos resultados se muestran en el anexo C.

4.2.1. Discusión de resultados

El resultado muestra la interconexión de 9 de cada 10 genes objetivos en el mismo módulo. Este resultado se consideró satisfactorio, ya que confirmó la hipótesis que indica que los genes glicolíticos de las semillas deben formar un módulo en el que estén cerca uno del otro. Sin embargo, no alcanzaron la misma modularidad aplicando el mismo método para obtener los mejores pares de predictores.

En este caso, se aplica el método de agrupamiento lineal para inferir la red glicolítica considerando los 10 mejores pares de predicción para cada semilla de la glicólisis como objetivo para evaluar su modularidad. La Figura 4.1 muestra el resultado.

Sorprendentemente, todos los genes objetivos están interconectados en el mismo componente conectado, lo que indica que la aplicación del agrupamiento lineal aumentó el poder de estimación sin perder la información significativa después del proceso de agrupación. De hecho, la tabla de distribución de probabilidad conjunta original contiene $3^2 = 9$ posibles

casos, mientras que el agrupamiento lineal asigna de estos 9 casos a 5 clases de equivalencia (los posibles valores de combinación lineal para los pares de predicción son genes ternarios $R = \{0, 1, 2\}$) estás son: (0, 1, 2, 3, 4). La tabla 4.1 muestra un resumen de los datos necesarios del método de agrupamiento lineal en comparación al método publicado por Barrera en [Barrera et al., 2007]. En conclusión podemos afirmar que el método de agrupamiento lineal obtuvo el mismo resultado que el método de barrera, utilizando tablas de distribución de probabilidades mas pequeñas (9 a 5).

Cuadro 4.1: Comparación articulo Barrera y método de agrupamiento lineal, para obtener la red glicolítica.

Parametros	Articulo Barrera	Agrupamiento lineal
Mejores predictores individuales	10	0
Mejores pares de predictores	5	10
Tamaño de tabla de probabilidad por pares	$3^2 = 9$	5 (clases de equivalencia)

Además de comprobar el poder de estimación del método, este resultado sugiere nuevos focos de estudio biológico en la red de regulación de la glicólisis, son interesantes casos de estudio aquellos genes que interconectan la red, es decir aquellos que trazan un camino entre los genes semilla, estos genes pueden ser tomados como objeto de estudio para entender la relación y colaboración entre los genes semillas.

La inferencia de redes genéticas a partir de datos de expresión es

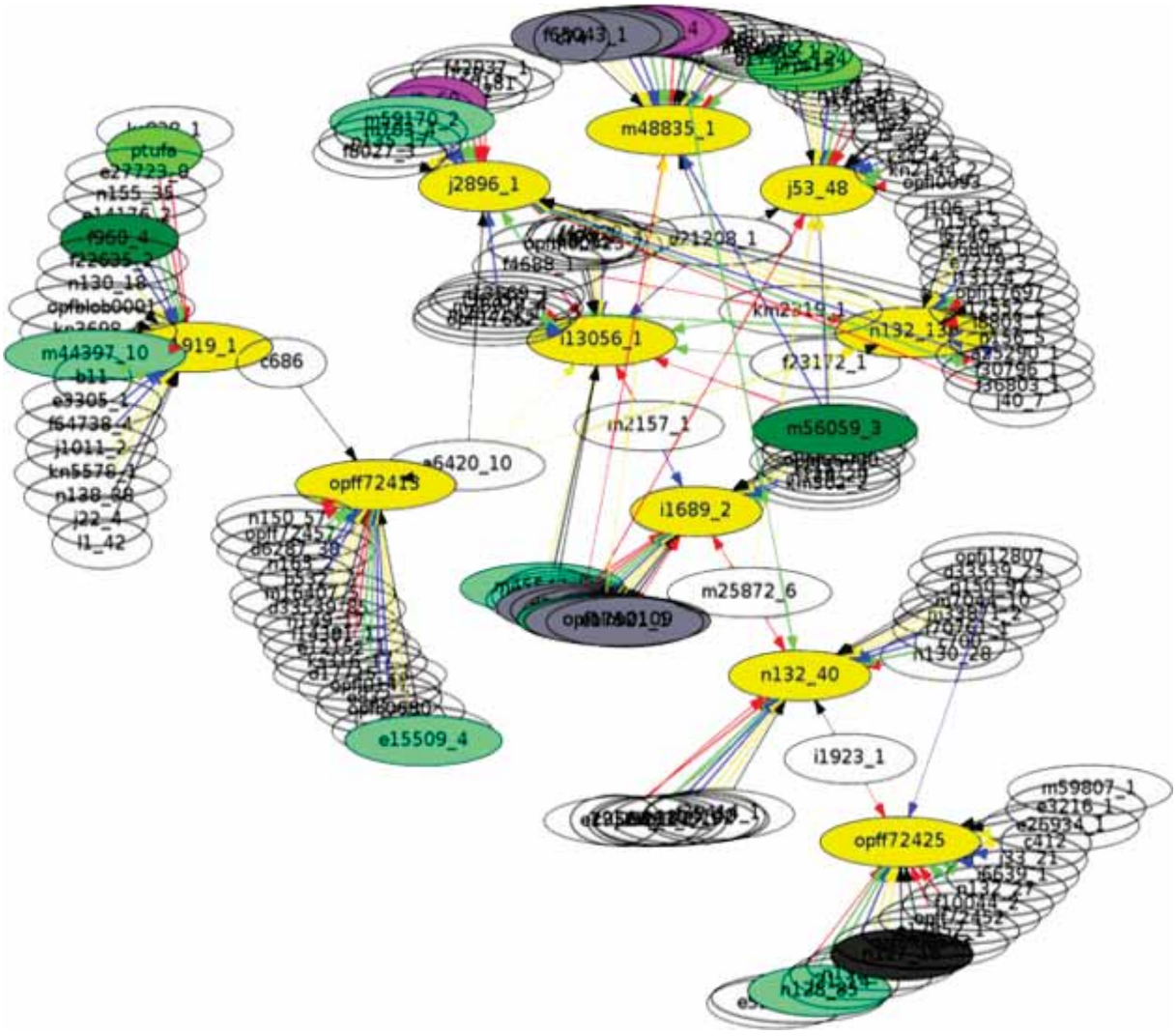


Figura 4.1: Red Glicolítica obtenida al aplicar el método de agrupamiento lineal para los mejores 10 pares predictores para cada gen de la semilla glicolítica. La semilla glicolítica esta representado de color amarillo.

Conclusiones

1. Con el fin de avanzar en el conocimiento biológico del *Plasmodium faciparum*, se construyó una red genética probabilística (PBN) estimada por el método de *agrupamiento lineal* con entropía condicional media, a partir de señales dinámicas de expresiones genéticas del estado asexual sanguíneo del parásito, publicados por Bozdech en [Bozdech and DeRisi, 2005], esta red mostró resultados potencialmente importantes desde el punto de vista biológico similares a los encontrados en [Barrera et al., 2004] y [Barrera et al., 2007]. Este resultado se consideró satisfactorio, porque confirma la efectividad del método para trabajar con datos reales, ya que la red construida muestra la hipótesis que indica que los genes glicolíticos deben formar un módulo en el que están cerca uno del otro, como se puede observar en la figura 4.1.

2. Para la construcción de la red glicolítica tomamos los 10 mejores pares de genes predictores de la vía glicolítica del *Plasmodium faciparum* como posibles predictores de los genes semilla, estos llegaron a mostrar un eficiente resultado ya que en los experimentos la información mutua de estos genes predictores que forman el puente entre los genes glicolíticos tiende a presentar los mejores resultados lo que refuerza la hipótesis de colaboración entre estos genes, y a su vez sugiere la importancia de estos genes y los ofrece como posibles focos de estudio.

3. Como se explicó en la sección 4.2.1, la aplicación del agrupamiento aumento el poder de estimación sin perder información significativa, pero aún tiene problemas en encontrar la modularidad de cada gen, esto reafirma lo dicho en [Montoya, 2014], que sugiere un próximo estudio donde se podría generar una estrategia híbrida que solo aplique el agrupamiento cuando sea conveniente, esta función de decisión es no

trivial y constituye un foco de estudio interesante que ayudaría a comprender el tipo de dependencia entre genes, es decir su poder de activación e inhibición.

Recomendaciones

1. Profundizar el análisis del método de *Agrupamiento lineal*, con las dos variaciones descritas de este modo podríamos mejorar los resultados obtenidos.

2. Aplicar la evaluación del método con otras funciones criterio, además de la entropía condicional media, por ejemplo podríamos adoptar el Coeficiente de determinación no lineal (CoD) (ver Sección 2.7.6).

3. Perfeccionar el método a través de un análisis multiresolución para aplicar el agrupamiento cuando sea necesario, de este modo se podría evitar la perdida de información que ocurre después del agrupamiento en las clases de equivalencia.

4. Explorar otros tipos de agrupamiento en las clases de equivalencia, que no necesariamente se basen en agrupamientos lineales.

5. Utilizar métodos auxiliares como el aprendizaje supervisado para que aprenda a calibrar adecuadamente la penalización envuelta en la función criterio.

Anexos

Anexo A

Muestras

Muestra # 1

1 1 2 2 2 2 2 1 2 2 2 1 2 1 2 2 2 2 0 2 2 1 0 1 2 0 1 0 0 0 2 0 2 1 0 1 0 2 0 2 1 0 0 2 1 0 1 0 1 0 1
2 2 1 0 1 1 1 2 2 2 0 0 0 1 1 2 0 1 1 0 1 1 2 1 0 2 1 0 0 0 1 0 2 1 1 1 1 1 1 1 1 1 1 2 1 1 2 1 2 2 2
0 0 0 1 2 0 0 0 2 1 2 1 1 0 2 2 2 2 1 1 2 1 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 2 0 0 1 2 1 2
2 1 0 1 1 1 1 0 2 2 2 2 1 2 1 1 0 0 1 0 0 1 1 1 1 0 1 1 1 1 1 2 0 1 1 1 1 1 1 0 1 0 1 2 1 1 1 2 2 2 2
1 1 0 2 2 1 1 0 0 1 0 1 0 2 2 0 2 1 1 2 0 2 0 0 0 1 1 0 1 1 1 1 1 1 1 0 2 1 0 0 0 1 2 2 2 2 1 0 1 0 2
1 1 2 0 2 2 2 0 2 0 1 1 2 2 2 1 2 1 2 1 1 0 1 0 0 1 0 2 2 1 1 1 0 2 0 0 0 2 1 1 1 1 1 1 1 0 0 1 1 1 1 0 0 1
1 1 1 1 1 0 0 0 1 1 2 0 2 1 1 0 1 1 1 1 2 1 1 1 0 1 1 2 1 0 0 1 1 2 2 2 1 2 1 0 0 1 0 1 1 0 0 1 2 0 1
1 1 1 0 1 0 2 0 1 1 1 1 0 1 0 1 1 2 1 1 0 2 1 2 2 1 2 1 0 1 0 0 2 0 1 2 0 0 1 1 1 1 1 1 1 1 0 0 0 1 1
1 2 2 2 2 2 0 1 0 2 0 1 1 0 0 2 1 1 1 0 0 0 2 1 1 0 2 0 1 1 2 2 1 0 2 2 0 0 1 1 0 1 1 0 1 1 0 0 1 2 2 0
2 1 1 1 1 1 1 2 0 1 2 1 0 1 0 2 0 2 0 1 1 1 2 2 1 2 1 1 1 1 2 2 0 0 1 2 1 2 2 2 1 1 1 1 1 1 1 1 1 1 1
2 1 0 0 2 1 1 2 2 1 1 1 1 0 0 2 1 1 1 2 1 1 2 2 1 1 0 0 0 0 1 1 2 1 2 2 2 1 2 2 1 1 0 1 2 1 1 1 0 1 1
1 2 2 2 2 1 0 1 1 0 0 0 0 1 1 2 1 1 0 1 1 1 1 1 2 2 1 0 1 1 2 2 1 0 2 2 1 1 1 0 2 0 1 1 1 0 0 1 0 1 0
0 0 0 0 0 1 1 1 2 2 2 2 0 2 0 2 2 2 1 2 2 1 1 0 1 1 0 0 0 1 1 2 0 2 1 1 2 0 1 1 0 0 1 0 0 0 0 1 2 1 1
2 2 2 2 2 0 1 1 0 1 2 2 2 2 2 0 0 0 2 0 1 1 1 1 1 1 2 2 1 2 0 0 1 2 2 2 1 1 1 2 1 1 1 1 2 1 1 0 1 2 2
1 0 2 0 2 2 1 0 1 2 1 1 1 1 1 0 2 1 2 1 2 1 0 0 2 2 1 2 1 1 0 1 1 1 2 0 0 1 1 1 2 2 2 2 1 0 2 1 1 2 1
0 1 0 0 2 1 1 2 2 0 1 1 2 1 2 2 1 1 0 0 2 1 2 1 2 2 0 0 1 0 1 1 0 2 2 1 1 0 0 2 0 0 1 2 1 1 1 2 0 2 1
1 2 2 0 0 2 0 1 1 0 1 1 1 2 0 2 2 1 1 2 1 1 1 1 1 0 0 0 2 2 2 2 2 0 1 0 0 1 1 0 1 1 0 0 1 0 1 1 0 0 0
0 2 1 1 1 2 2 1 1 1 1 2 0 1 0 1 1 1 1 0 1 1 1 2 2 1 2 1 1 1 1 1 1 1 2 0 0 0 1 1 1 2 1 1 1 0 1 0 0 0 1
1 0 1 0 1 1 1 1 1 1 1 0 1 0 1 1 2 1 2 1 0 0 0 0 0 1 1 1 2 1 1 0 2 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0
0 1 0 2 1 1 1 0 2 1 2 1 0 1 1 2 1 1 1 1 1 1 1 1 1 0 1 1 2 1 2 0 0 2 1 1 2 0 1 1 1 1 1 1 2 0 2 1 0 1 2 2

1 1 1 1 2 1 1 2 0 0 1 2 1 1 0 1 2 1 1 1 0 1 2 1 1 1 2 1 0 1 1 2 1 2 1 1 1 1 0 0 0 0 2 1 0 1 0 1 0 1 1
2 0 2 2 1 1 0 2 0 1 0 2 1 0 2 0 2 1 1 2 0 2 1 2 0 1 1 0 1 2 2 2 2 2 2 1 2 2 2 2 0 0 0 1 2 0 1 1 0 2 0 1
1 2 1 0 2 1 1 0 1 0 0 0 2 0 0 0 1 0 2 1 2 2 2 1 1 0 1 0 0 0 1 0 0 0 0 2 1 1 0 0 2 1 0 0 1 0 1 1 2 0 0
1 2 2 0 1 2 2 1 2 1 1 1 0 0 1 1 0 1 0 1 1 1 1 0 0 2 1 1 2 2 2 2 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 2 2 0 2
1 2 1 1 2 2 0 1 0 0 0 1 1 1 2 0 2 2 1 2 0 2 2 1 1 1 1 1 0 1 0 1 2 1 1 1 2 0 0 0 2 0 1 0 1 1 0 0 0 1 1
0 0 0 0 0 2 1 1 1 0 0 0 1 1 2 0 0 0 1 1 1 1 0 1 1 0 1 0 1 1 1 2 1 0 1 1 0 2 1 0 0 2 2 1 2 0 1 1 1 0 1
1 0 2 2 1 2 0 0 0 2 1 1 1 1 1 1 1 1 0 0 0 2 1 2 1 1 1 2 2 0 2 0 0 2 0 0 0 1 1 1 1 0 1 1 0 1 2 2 1 0 0
2 1 2 1 2 1 0 2 1 0 2 2 1 1 2 1 1 1 2 0 1 0 1 0 2 1 2 1 1 1 2 0 1 1 0 1 0 2 1 2 2 0 2 2 0 2 0 1 0 1 1
0 0 1 1 1 1 0 2 1 0 1 1 2 2 1 2 1 1 1 1 2 1 1 1 2 0 2 1 1 0 1 1 0 1 2 0 0 0 1 0 1 0 1 2 1 2 0 2 1 2 0
0 1 1 1 2 0 0 2 1 2 2 1 0 2 1 1 2 2 1 1 1 2 0 0 1 1 2 2 1 1 2 2 1 0 1 2 1 2 1 0 1 0 1 0 1 1 1 0 2 2 1
1 1 0 0 1 2 2 1 1 2 0 0 1 1 0 0 2 2 1 1 1 2 2 1 1 0 1 1 1 1 0 0 2 2 1 1 2 2 1 1 2 2 1 1 0 1 1 1 0 0 1
1 1 2 2 0 0 0 1 0 0 2 0 0 1 2 0 0 1 1 2 0 2 1 1 0 1 0 1 2 1 2 1 1 1 0 2 1 1 2 1 2 1 1 1 0 0 2 0 1 2 0
2 1 2 0 1 1 1 1 1 2 2 1 0 1 1 1 1 0 0 0 0 0 1 2 2 2 1 2 1 1 1 2 0 2 1 2 2 1 0 1 1 2 0 2 2 2 2 1 1 2 0
0 1 1 1 1 0 1 0 0 1 1 1 1 2 1 1 0 1 2 0 0 0 0 0 2 2 1 1 0 1 0 2 1 2 2 2 1 2 2 2 1 1 2 1 1 2 2 2 1 0 0
1 1 1 2 2 2 2 2 2 0 1 2 2 1 1 1 2 2 1 1 1 1 0 1 1 1 0 1 0 0 1 0 1 1 1 2 1 1 1 0 1 0 1 1 2 2 1 2 1 1 1 0
2 1 0 1 1 1 0 0 1 0 1 1 0 1 1 1 0 0 1 0 2 1 1 2 2 1 1 1 1 0 1 1 2 1 1 0 0 2 2 2 2 0 1 0 1 1 0 0 0 1 1 0
1 1 2 0 1 1 0 2 0 0 1 1 0 1 0 1 1 1 1 0 2 0 1 1 1 1 1 1 0 1 1 1 2 1 2 2 1 1 2 0 1 0 0 0 1 0 1 1 2 1 1 1
1 0 0 1 1 1 2 1 1 2 1 1 1 2 1 1 0 0 1 1 2 2 1 0 0 1 2 2 2 2 0 0 0 1 1 0 1 2 0 0 1 2 1 1 1 0 1 0 0 1 0
0 1 1 1 2 1 2 1 0 2 1 0 1 2 1 2 2 2 2 0 2 2 0 1 2 1 1 1 0 0 2 1 1 2 2 1 2 2 0 1 2 0 2 2 0 0 1 1 1 2 2 1
0 0 2 1 2 2 0 0 0 0 0 1 1 2 1 1 2 2 1 2 1 0 1 1 1 1 1 1 1 1 2 2 2 2 2 0 1 0 2 2 1 2 1 1 1 1 1 1 1 2 1 0 0
2 0 1 1 2 1 1 1 1 0 0 1 2 0 0 2 1 2 0 2 1 0 2 2 2 2 2 2 1 2 2 2 2 1 2 1 0 1 1 0 1 1 1 1 1 1 1 1 2 1 1 1
1 1 1 0 0 1 1 1 2 1 0 2 1 1 2 2 1 0 2 2 1 1 1 0 1 2 0 0 2 0 1 2 1 2 2 0 0 1 1 2 2 2 1 2 2 1 2 1 2 1 1
2 1 1 1 0 2 1 1 2 2 1 2 1 1 1 1 2 2 1 1 1 0 0 1 1 1 2 0 0 1 0 0 1 0 2 0 2 0 0 0 0 0 0 1 1 2 0 1 1 2 1
1 1 1 2 2 2 1 2 0 1 0 2 1 1 0 1 1 1 2 2 1 0 2 1 1 1 2 2 2 1 0 2 1 0 1 1 2 0 0 1 1 1 0 1 0 2 2 1 1 0 1
0 1 1 1 1 1 0 1 1 1 1 1 0 1 2 1 1 1 0 1 0 1 2 0 1 2 2 2 0 1 1 2 2 1 1 1 1 2 1 1 2 2 0 1 1 1 2 1 1 0 1
1 0 1 1 1 2 2 2 2 2 2 1 2 1 1 0 0 1 0 0 1 2 1 0 1 0 1 2 2 1 0 0 1 1 1 1 2 0 2 2 1 1 1 0 1 1 1 0 1 1 0
2 0 1 1 1 2 1 1 1 2 1 0 0 1 0 1 1 2 2 1 1 1 2 1 2 1 1 1 2 2 1 2 1 1 0 2 0 0 1 1 0 1 1 1 0 1 0 2 2 2 1 2
1 1 1 2 1 2 2 2 2 1 0 1 0 1 1 2 2 2 2 2 1 2 2 2 1 2 2 1 0 1 1 2 2 1 1 1 1 1 1 1 1 2 1 0 1 2 0 1 1 2 1 1 2 1 2 1 2 1
1 1 1 0 1 1 1 2 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 2 2 2 2 0 0 1 1 1 1 2 0 0 0 0 0 2 1 2 2 0 0
2 1 2 2 2 1 0 1 1 1 1 1 1 0 1 1 2 0 1 1 1 1 2 2 1 1 0 1 1 0 0 2 1 1 1 1 0 1 2 1 0 1 0 1 1 0 1 0 1 0 0
1 0 2 0 1 0 2 1 1 1 2 2 1 2 2 1 1 1 0 1 1 1 1 2 1 2 0 1 1 1 1 0 1 1 2 1 2 0 2 0 1 2 0 0 0 0 2 2 2 2 2
1 2 2 1 2 2 0 1 2 1 1 2 0 1 0 2 2 1 1 2 0 2 2 1 1 1 1 1 1 1 0 2 0 1 0 1 1 1 0 1 2 1 1 2 0 1 1 1 0 2 1
0 0 1 0 1 2 1 1 1 0 1 1 1 0 1 2 1 1 2 1 1 1 2 1 0 2 2 1 1 1 2 1 2 1 1 1 1 2 1 2 0 2 2 0 1 1 0 1 1 1 1

1 1 1 1 1 1 2 2 0 0 0 0 2 2 2 0 2 1 1 0 1 1 0 2 0 2 2 2 2 1 2 2 2 1 1 1 2 1 1 1 2 1 1 2 1 1 1 1 1 0 1
1 1 2 2 0 1 1 1 1 1 1 2 1 2 1 0 2 2 0 1 1 0 1 0 1 1 0 0 0 2 1 1 2 2 0 0 1 1 2 1 2 0 1 1 2 1 1 1 1 1 2
1 2 0 2 2 1 2 2 2 2 1 1 2 2 1 1 2 2 2 0 1 1 0 0 2 1 2 0 1 1 2 2 2 2 2 1 1 1 1 1 2 2 1 0 0 2 2 1 1 1 1
0 0 1 2 1 1 1 1 1 2 2 0 2 2 1 0 2 2 2 2 1 2 2 0 0 1 2 0 2 1 0 2 2 1 2 2 2 0 0 1 1 2 2 1 1 1 1 1 1 2 1 2
2 1 2 2 1 1 1 1 1 1 1 2 1 1 1 1 1 0 1 1 0 1 2 1 2 1 0 1 1 1 2 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1 1 1 1 0 1
2 1 1 1 1 1 2 2 2 0 1 1 1 0 0 1 1 1 2 1 0 1 1 1 1 2 1 0 1 1 1 2 2 2 2 2 0 1 1 1 0 1 1 2 1 1 0 1 1 1 2
1 0 1 0 2 1 2 1 0 1 0 0 1 2 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 0 0 1 2 1 1 1 0 2 1 1 2 1 1 1 1 2 1 1 2 2 2 0
2 2 2 2 1 1 1 0 1 2 1 1 2 1 1 0 2 2 2 1 1 1 1 1 1 1 0 2 1 1 0 2 2 2 2 2 2 1 1 2 1 0 1 2 2 2 1 2 0 2 2 1 0
1 2 0 0 1 0 0 1 1 2 2 0 1 1 2 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 2 2 2 2 1 2 2 0 2 0 1 1 2 0 2 2 2 1 1 2 2
1 1 1 1 1 2 0 2 0 1 1 1 1 2 2 2 2 1 1 2 2 2 1 2 2 2 2 2 2 1 1 1 1 0 0 1 0 0 1 1 1 2 1 1 1 2 1 0 0 1 0 1
1 1 0 1 1 0 0 1 1 1 2 1 2 2 2 0 1 0 2 1 2 2 2 1 2 2 1 1 0 0 1 1 1 0 2 2 1 2 1 2 1 1 1 1 1 0 1 0 1 1 1
1 1 1 1 1 1 1 1 1 0 2 0 1 1 2 1 2 1 0 1 1 1 2 2 2 1 1 2 0 0 1 0 1 1 0 0 1 1 2 1 1 0 0 1 0 2 2 1 1 2 0 2
0 1 0 0 1 1 1 1 2 1 1 1 1 2 1 1 0 0 1 1 2 1 1 2 0 2 2 1 1 2 2 1 2 1 1 1 1 0 2 1 1 0 2 2 1 1 0 0 1 1 0
1 2 1 0 0 0 0 1 0 0 0 0 2 2 1 2 1 2 1 2 2 0 1 1 1 1 2 0 2 1 1 2 0 1 0 2 2 0 2 2 0 1 0 0 2 1 0 1 0 0 2
1 2 1 1 0 2 2 2 0 1 1 1 1 1 1 1 2 0 2 1 0 1 1 1 1 0 1 1 1 0 0 2 2 1 1 1 1 1 1 2 1 1 2 2 2 2 1 1 1 1 1
2 1 2 2 2 1 1 1 1 0 1 1 1 1 1 1 2 2 1 1 1 2 0 0 0 2 0 1 0 2 2 1 1 1 1 2 1 1 1 2 2 2 0 1 2 1 1 1 1 2 2
1 1 2 0 0 2 1 1 1 2 2 1 1 1 0 2 1 1 1 1 1 1 1 2 2 2 2 1 1 0 1 2 1 1 0 0 1 1 1 1 2 1 1 2 1 1 1 0 1 2 0 2 1
1 0 1 2 1 1 2 1 2 2 0 0 0 1 1 1 1 0 0 1 0 0 0 2 2 1 1 1 0 2 0 2 0 2 1 1 0 1 1 0 1 1 2 2 2 1 1 2 0 1 1
0 1 1 0 0 1 1 1 1 1 1 1 1 1 0 1 0 1 2 1 0 1 0 0 1 1 1 1 1 1 0 0 1 1 0 0 0 2 1 1 1 2 1 2 2 1 0 1 2 1 1 1 1
2 2 2 2 1 1 1 1 0 0 1 1 0 1 1 1 2 1 1 1 2 0 0 2 0 2 2 2 0 1 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1
2 1 1 1 0 2 0 1 2 0 1 1 0 0 1 2 2 0 0 2 0 0 0 1 1 2 1 2 1 0 1 1 1 1 0 1 1 0 1 0 1 2 1 2 0 2 1 2 1 2 2
1 0 0 2 1 1 2 1 2 1 2 2 2 1 1 1 1 1 1 2 2 0 2 0 0 0 0 0 0 1 1 1 0 0 1 2 0 0 0 0 1 0 0 0 2 1 2 2 1 2 1
1 1 2 2 1 2 1 1 1 1 1 0 1 2 0 1 2 2 1 1 1 0 1 0 1 0 2 0 1 2 1 2 1 0 1 2 2 1 2 1 1 1 1 2 1 1 0 1 1 2 1
1 2 2 2 0 0 1 0 1 0 1 1 2 2 2 1 2 1 1 0 1 1 1 1 1 0 1 0 2 1 2 0 1 0 2 1 1 2 1 2 2 2 0 0 1 2 1 0 1 1 1 2
2 1 2 2 0 2 2 1 1 1 2 1 1 0 1 2 1 1 2 1 0 1 1 1 1 2 2 2 1 2 1 1 1 1 2 0 0 1 2 1 2 1 1 1 1 1 1 2 2 0 2
1 2 0 1 2 1 1 2 1 1 1 0 1 1 1 1 1 1 1 1 0 1 2 0 0 1 1 1 1 1 1 1 1 1 1 2 2 1 0 1 1 1 0 1 1 1 0 0 0 1 0 0
0 1 1 1 0 0 1 0 1 1 1 1 2 1 2 1 1 1 1 0 1 1 2 1 0 1 1 1 1 0 2 1 1 1 1 0 0 2 1 1 1 1 1 1 1 0 2 2 0 0 1 0
0 1 2 2 1 1 0 1 1 2 1 2 2 2 0 1 2 2 1 1 2 1 1 1 1 1 1 1 0 1 1 2 1 0 0 1 1 1 1 1 2 1 2 1 0 2 2 1 2 2 1 2 2
0 0 1 0 2 0 1 1 1 1 2 1 1 0 0 1 1 1 2 1 2 2 0 2 2 1 1 2 2 2 2 2 1 1 2 1 1 1 1 1 0 2 0 2 2 2 1 1 1 0 1
1 1 1 1 1 0 1 1 2 2 2 1 0 1 1 1 0 0 0 0 1 1 2 1 0 1 1 1 1 1 0 0 1 0 1 1 1 0 1 0 2 0 2 0 0 1 0 1 0 2 1
0 0 1 2 0 0 0 0 1 0 0 2 2 1 1 2 0 0 0 1 0 0 1 2 0 2 1 2 0 1 2 1 2 2 2 1 0 1 1 1 1 1 1 2 2 1 1 1 2 0 1
1 0 0 1 2 2 0 2 2 0 1 2 2 0 1 2 2 1 0 1 2 1 0 0 1 2 1 1 1 2 1 1 1 1 1 1 1 2 1 2 1 1 0 1 1 1 1 1 1 1 1
0 2 1 2 0 2 1 2 2 2 1 0 0 1 1 2 2 1 1 1 2 2 1 2 1 1 1 0 0 1 2 0 1 2 2 1 2 1 2 1 1 0 2 0 2 0 1 0 1 1 1

1 2 1 1 0 2 2 1 2 1 2 0 2 1 2 0 2 2 2 2 0 1 1 1 1 2 2 1 1 2 0 2 2 1 1 1 2 2 1 1 1 1 2 2 1 2 1 1 0 1 2
2 2 1 2 1 2 1 1 1 0 1 0 1 1 1 1 2 1 0 0 2 1 1 2 1 0 2 2 0 0 1 1 2 1 2 0 2 1 1 1 2 2 2 1 0 1 0 1 1 2 1
1 2 1 1 0 1 1 0 2 2 2 2 1 2 0 0 1 1 1 0 1 0 1 1 1 1 0 1 2 2 2 1 2 1 1 1 1 1 2 1 0 0 1 1 2 1 1 0 0 2 1
1 1 2 2 1 2 0 1 0 2 2 0 0 1 1 2 1 0 0 0 1 1 1 1 2 1 1 1 1 0 1 1 2 0 1 1 2 2 2 1 0 1 2 1 1 1 0 2 2 0 1
1 1 1 2 2 0 1 1 2 1 1 1 1 0 1 0 2 2 2 2 1 1 0 1 1 1 1 0 2 2 1 1 2 2 2 2 0 1 2 2 1 0 0 0 1 1 2 1 2 1 1
1 1 1 1 1 1 0 1 1 2 1 1 2 1 1 1 1 0 1 0 2 1 0 1 2 0 0 1 1 1 1 1 0 1 1 1 1 1 0 0 0 2 1 1 1 1 0 1 1 2 0
1 0 2 1 1 1 1 0 1 1 0 2 1 1 1 2 2 2 1 2 0 2 0 1 1 1 1 1 2 0 2 1 1 1 1 0 1 1 1 2 1 1 2 2 1 1 2 0 1 2 1
2 1 2 1 0 1 1 1 2 1 1 1 2 2 1 2 1 1 1 1 1 1 0 1 1 1 1 2 2 1 1 0 1 1 1 2 2 2 2 2 1 2 2 1 2 1 1 2 1 1 1
1 2 1 2 1 2 1 2 0 2 0 2 1 1 1 2 1 2 1 2 1 1 1 1 1 1 1 2 1 1 0 1 1 1 1 0 1 2 1 2 0 2 1 1 1 2 1 1 0 1 2 2
1 0 1 1 1 1 1 0 0 0 2 2 2 1 1 1 0 0 2 1 1 1 2 1 0 1 1 0 0 1 1 1 1 0 0 2 1 1 2 1 1 1 1 1 1 2 1 1 2 0 2
2 1 2 2 0 0 1 1 1 1 2 1 2 2 2 1 2 2 2 1 0 2 1 1 1 1 0 2 1 1 1 1 1 0 1 1 0 1 1 2 1 1 2 2 1 2 1 1 1 2 1
2 0 2 1 0 1 0 2 1 2 1 1 1 1 1 0 0 1 1 1 1 2 1 1 2 2 0 1 2 1 1 1 0 2 1 0 0 1 0 1 1 0 1 1 0 1 0 2 1 1 1
2 2 1 2 0 2 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 2 2

Anexo B

Código Fuente

```
/*
 * CLASE GenPredictors
 */
package Inference;

import Inference.InferenceNetwork.Tipo_Inferencia;
import Networks.MicroArray;
import java.io.Serializable;
import java.util.Arrays;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Objects;
import java.util.TreeSet;
import Resources.Resources;

/**
 * Almacena datos de la prediccion: NroGen, Arreglo de predictores, Ganancia de
     Informacion
 */
public class GenPredictors implements Comparable<GenPredictors>, Serializable {

    public transient static int[][][] definitionGroup = null;
    public transient static int sizeGroup=-1;
    protected int gen;
```

```
protected TreeSet<Integer> predictors;
protected long[] TruthTable;
protected InferenceNetwork.Penalizacao penalty;
protected double mutualInformation;
protected MicroArray microArray;
protected Tipo_Inferencia typeOfInference;
protected int groupingSelected;
private boolean removeNoise;

/**
 *
 * @param gen
 * @param predictors
 * @param penalty
 * @param microArray
 * @param typeOfInference
 */
public GenPredictors(int gen, TreeSet<Integer> predictors,
    InferenceNetwork.Penalizacao penalty, MicroArray microArray, Tipo_Inferencia
    typeOfInference) {
    this.gen = gen;
    this.predictors = predictors;
    this.penalty = penalty;
    this.microArray = microArray;
    this.typeOfInference = typeOfInference;
    if (this.typeOfInference!= Tipo_Inferencia.WITHOUT_GROUPING &&
        !predictors.isEmpty() && predictors.size() != sizeGroup) {
        sizeGroup=predictors.size();
        definitionGroup=defineGroups();
    }
    removeNoise = true;
}

public int getGen() {
    return gen;
}
```

```
public TreeSet<Integer> getPredictors() {
    return predictors;
}

/* @param removeNoise */
public void setRemoveNoise(boolean removeNoise) {
    this.removeNoise = removeNoise;
}

public long[] getTruthTable() {
    return TruthTable;
}

public Tipo_Inferencia getTypeOfInference() {
    return typeOfInference;
}

public InferenceNetwork.Penalizacao getPenalty() {
    return penalty;
}

public double getMutualInformation() {
    return mutualInformation;
}

public MicroArray getMicroArray() {
    return microArray;
}

/**
 * Determina el indice en la tabla de predictores para la instacia noInstances.
 * @param noInstances nro de instancia de tiempo para recuperar el indice
 * @return indice en la tabla de predicciones
 */
public int IndexTruthTable(int noInstances) {
    int no = 0;
    int i = 0;
```

```
    for (Iterator<Integer> it = predictors.iterator(); it.hasNext(); i++) {
        int preditor = it.next();
        no += ((microArray.getData()[preditor][noInstances]) ? (1 << i) : 0);
    }
    return no;
}

/**
 * Calcula el ganho de informacion del gen para los predictors, almacena el valor
     en el atributo ganhoInformacion y retorma la tabela de verdad
 * @param entropyTable
 */
public void assignMutualInformation(int[][] entropyTable) {
    //percurso no gen no tempo t e nos predictors tempo t+1
    switch (typeOfInference) {
        case WITHOUT_GROUPING: {
            groupingSelected = -1;
            this.mutualInformation = calculateMutualInformation(entropyTable);
        }
        break;
        case GROUPING_LINEAR: {
            int[][][] table = createTableGrouping(entropyTable);
            double information = Double.NEGATIVE_INFINITY;
            groupingSelected = -1;
            for (int i = 0; i < table.length; i++) {
                int[][] ises = table[i];
                double estaInfo = calculateMutualInformation(ises);
                if (estaInfo > information) {
                    groupingSelected = i;
                    information = estaInfo;
                }
            }
            this.mutualInformation = information;
        }
        break;
        case GROUPING_SELECTIVE: {
            if (needGrouping(entropyTable)) {
```

```
                this.typeOfInference = Tipo_Inferencia.GROUPING_LINEAR;
            } else {
                this.typeOfInference = Tipo_Inferencia.WITHOUT_GROUPING;
            }
            assignMutualInformation(entropyTable);
        }
        break;
        case GROUPING_JOINT: {
            groupingSelected = -1;
            double information = calculateMutualInformation(entropyTable);
            int[][][] tables = createTableGrouping(entropyTable);
            for (int i = 0; i < tables.length; i++) {
                int[][] ises = tables[i];
                double estaInfo = calculateMutualInformation(ises);
                if (estaInfo > information) {
                    groupingSelected = i;
                    information = estaInfo;
                }
            }
            this.mutualInformation = information;
        }
        break;
    }
}

/**
 * @param entropyTable
 * @return
 */
private double calculateMutualInformation(int[][] entropyTable) {
    double alpha = 0.5;
    //double[] entropiasParciais = new double[entropyTable.length];
    double entropyYX = 0;
    //contagem de 0s y 1s
    int[] withAll = new int[]{0, 0};
    //calcular H(Y) entropy de Y={0,1} total
    for (int iesimaLine = 0; iesimaLine < entropyTable.length; iesimaLine++) {
```

```
        withAll[0] += entropyTable[iesimaLine][0];
        withAll[1] += entropyTable[iesimaLine][1];
    }
    double entropyAll = Resources.entropy((double) withAll[0] /
        (microArray.getNroExperiments() - 1), 2)
            + Resources.entropy((double) withAll[1] /
                (microArray.getNroExperiments() - 1), 2);
    if (entropyAll != 0) {
        //recorer tabla y calcular entropias parciales
        for (int iesimoLine = 0; iesimoLine < entropyTable.length; iesimoLine++) {
            double entropyPartial = 0;
            //nro de predicciones por linea
            int sum = entropyTable[iesimoLine][0] + entropyTable[iesimoLine][1];
            //calcular H(Y|x) entropy de Y dado el estado x
            if (sum != 0) {
                entropyPartial =
                        Resources.entropy((double) entropyTable[iesimoLine][0] /
                            sum, 2)//entropia H(Y|x=0)
                        + Resources.entropy((double) entropyTable[iesimoLine][1] /
                            sum, 2);//entropia H(Y|x=1)
            } else if (penalty == InferenceNetwork.Penalizacao.NOT_OBSERVED) {//
                Penaliza instancias nao observadas pega a entropy total,
                entropyPartial = entropyAll;
            }
            //calcular P(x)H(Y|x) la entropy por la probabilidad,ou
                (f(x)+alpha)*H(Y|x)
            entropyPartial *= (penalty ==
                InferenceNetwork.Penalizacao.WITHOUT_PENALTY)
                    ? sum : (sum + alpha);
            // ((microArray.getNroExperiments() - 1) + (alpha *
                entropyTable.length)));
            entropyYX += entropyPartial;
        }
        //calcular H(Y|X) entropy de Y para todas las X, ou sum((f(x)+alpha)*H(Y|x))
        entropyYX /= (penalty == InferenceNetwork.Penalizacao.WITHOUT_PENALTY)
                ? (double) (microArray.getNroExperiments() - 1)
```

```
                : (double) ((alpha * entropyTable.length) +
                    (microArray.getNroExperiments() - 1));

        return entropyAll - entropyYX;
    } else {
        return 0;
    }
}

double entropyGen() {
    int[][] tableEntropy=createTable();
    int[] withAll = new int[]{0, 0};
    //calcular H(Y) entropy de Y={0,1} total
    for (int iesimoLine = 0; iesimoLine < tableEntropy.length; iesimoLine++) {
        withAll[0] += tableEntropy[iesimoLine][0];
        withAll[1] += tableEntropy[iesimoLine][1];
    }
    return Resources.entropy((double) withAll[0] / (microArray.getNroExperiments()
        - 1), 2)
            + Resources.entropy((double) withAll[1] /
                (microArray.getNroExperiments() - 1), 2);
}

public class LineEntropy {

    int fy0;
    int fy1;
    double HYx;
}

/**
 * @param tableEntropy
 * @return
 */
protected LineEntropy[] TableMutualInformation(int[][] tableEntropy) {
    LineEntropy[] lineEntropies = new LineEntropy[tableEntropy.length + 1];
    int alpha = 1;
```

```
//double[] entropiasParciais = new double[tableEntropy.length];
double entropyYX = 0;
//contagem de 0s y 1s
int[] withAll = new int[]{0, 0};
//calcular H(Y) entropy de Y={0,1} total
for (int iesimaLine = 0; iesimaLine < tableEntropy.length; iesimaLine++) {
    withAll[0] += tableEntropy[iesimaLine][0];
    withAll[1] += tableEntropy[iesimaLine][1];
    lineEntropies[iesimaLine] = new LineEntropy();
}
lineEntropies[tableEntropy.length] = new LineEntropy();
lineEntropies[tableEntropy.length].fy0 = withAll[0];
lineEntropies[tableEntropy.length].fy1 = withAll[1];
double entropyAll = Resources.entropy((double) withAll[0] /
    (microArray.getNroExperiments() - 1), 2)
        + Resources.entropy((double) withAll[1] /
            (microArray.getNroExperiments() - 1), 2);

//recorer tabla y calcular entropias parciales
for (int iesimoLine = 0; iesimoLine < tableEntropy.length; iesimoLine++) {
    double entropyPartial = 0;
    //nro de predicciones por linea
    int sum = tableEntropy[iesimoLine][0] + tableEntropy[iesimoLine][1];
    //calcular H(Y|x) entropy de Y dado el estado x
    if (sum != 0) {
        entropyPartial =
                Resources.entropy((double) tableEntropy[iesimoLine][0] / sum,
                    2)//entropia H(Y|x=0)
                + Resources.entropy((double) tableEntropy[iesimoLine][1] / sum,
                    2);//entropia H(Y|x=1)
    } else if (penalty == InferenceNetwork.Penalizacao.NOT_OBSERVED) {//
        Penaliza instancias nao observadas pega a entropy total,
        entropyPartial = entropyAll;
    }
    lineEntropies[iesimoLine].fy0 = tableEntropy[iesimoLine][0];
    lineEntropies[iesimoLine].fy1 = tableEntropy[iesimoLine][1];
    lineEntropies[iesimoLine].HYx = entropyPartial;
```

```
            //calcular P(x)H(Y|x) la entropy por la probabilidad,ou (f(x)+alpha)*H(Y|x)
            entropyPartial *= (penalty == InferenceNetwork.Penalizacao.WITHOUT_PENALTY)
                    ? sum : (sum + alpha);
            //calcular H(Y|X) entropy de Y para todas las X, ou sum((f(x)+alpha)*H(Y|x))
            entropyYX += entropyPartial;
        }
        //calcular H(Y|X) entropy de Y para todas las X, ou sum((f(x)+alpha)*H(Y|x))
        entropyYX /= (penalty == InferenceNetwork.Penalizacao.WITHOUT_PENALTY)
                ? (double) (microArray.getNroExperiments() - 1)
                : (double) ((alpha * tableEntropy.length) +
                    (microArray.getNroExperiments() - 1));
        lineEntropies[tableEntropy.length].HYx = entropyAll;// - entropyYX;

        return lineEntropies;
    }

    public int[][] createTable() {
        //percurso no gen no tempo t e nos predictors tempo t+1
        int[][] tableEntropy = new int[1 << predictors.size()][];
        //inicializar a tabela
        for (int i = 0; i < tableEntropy.length; i++) {
            tableEntropy[i] = new int[]{0, 0};
        }
        //contagem de 0s y 1s
        for (int j = 0; j < microArray.getNroExperiments() - 1; j++) {
            tableEntropy[IndexTruthTable(j)][(microArray.getData()[gen][j + 1]) ? 1 :
                0]++;
        }
        return tableEntropy;
    }

    /**
     * @param nroPreditores
     */
    private int[][] defineCoefficients() {
        int noCoef = 1 << (predictors.size() - 1);//2 al nro de predites -1
        int[][] coefficients = new int[noCoef][predictors.size()];
```

```
        for (int i = 0; i < noCoef; i++) {
            int rest = i;
            for (int j = 0; j < predictors.size(); j++) {
                coefficients[i][j] = (rest % 2 == 0) ? 1 : -1;
                rest /= 2;
            }
        }
        return coefficients;
    }

    private int[][][] defineGroups() {
        int[][][] groups = new int[1 << (predictors.size() - 1)][predictors.size() +
            1][];
        //iniciar groups
        for (int i = 0; i < groups.length; i++) {
            for (int j = 0; j < groups[i].length; j++) {
                groups[i][j] = new int[Resources.noCombinations(predictors.size(), j)];
            }
        }
        //definir coefficients
        int[][] groupCoefficients = defineCoefficients();
        for (int iesimoGrouping = 0; iesimoGrouping < groupCoefficients.length;
            iesimoGrouping++) {
            //indices de groups
            int[] indexGroups = new int[predictors.size() + 1];
            //recuperar coefficients
            int[] coefficients = groupCoefficients[iesimoGrouping];
            //calcular el valor del primero grupo para desplasar a la posicion 0 en el
                arreglo
            int base = 0;
            for (int i = 0; i < coefficients.length; i++) {
                base += (coefficients[i] < 0) ? coefficients[i] : 0;
            }
            //evaluar y colocar cada indice en su lugar
            int sizeTable = 1 << predictors.size();
            for (int i = 0; i < sizeTable; i++) {
                int val = 0;
```

```
            for (int j = 0; j < coefficients.length; j++) {
                val += (((1 << j) & (i)) != 0) ? coefficients[j] : 0;
            }
            groups[iesimoGrouping][val - base][indexGroups[val - base]] = i;
            indexGroups[val - base]++;
        }
    }
    return groups;
}

/**
 *
 * @param table
 * @return
 */
public boolean needGrouping(int[][] table) {
    for (int i = 0; i < table.length; i++) {
        if (table[i][0] + table[i][1] == 0) {
            return true;
        }
    }
    return false;
}

private int[][][] createTableGrouping(int[][] table) {
    int[][][] tables = new int[1 << (predictors.size() - 1)][predictors.size() +
        1][2];
    for (int iesimoGrouping = 0; iesimoGrouping < definitionGroup.length;
        iesimoGrouping++) {
        for (int iesimoGroup = 0; iesimoGroup <
            definitionGroup[iesimoGrouping].length; iesimoGroup++) {
            for (int iesimoElement = 0; iesimoElement <
                definitionGroup[iesimoGrouping][iesimoGroup].length;
                iesimoElement++) {
                tables[iesimoGrouping][iesimoGroup][0] +=
                    table[definitionGroup[iesimoGrouping][iesimoGroup][iesimoElement]][0];
```

```
                tables[iesimoGrouping][iesimoGroup][1] +=
                    table[definitionGroup[iesimoGrouping][iesimoGroup][iesimoElement]][1];
            }
        }
    }
    return tables;
}

public void viewTableGrouping() {
    int[][][] tables = createTableGrouping(createTable());
    for (int j = 0; j < tables.length; j++) {
        int[][] ises = tables[j];
        System.out.println("Agrup " + j + ";");
        LineEntropy[] tabelaEntropia = TableMutualInformation(ises);
        for (int i = 0; i < tabelaEntropia.length; i++) {
            System.out.println(i + ";" + tabelaEntropia[i].fy0 + ";" +
                tabelaEntropia[i].fy1 + ";" + tabelaEntropia[i].HYx + ";");
        }
        System.out.println();
    }
}

public void viewTable() {
    System.out.println(toString());

    LineEntropy[] tabelaEntropia = TableMutualInformation(createTable());
    for (int i = 0; i < tabelaEntropia.length; i++) {
        System.out.println(i + ";" + tabelaEntropia[i].fy0 + ";" +
            tabelaEntropia[i].fy1 + ";" + tabelaEntropia[i].HYx + ";");
    }
}

/**
 *
 * @param predictorsEntropy
 * @return
 */
```

```
@Override
public int compareTo(GenPredictors predictorsEntropy) {
    if (mutualInformation > predictorsEntropy.mutualInformation) {
        return -1;
    }
    if (mutualInformation < predictorsEntropy.mutualInformation) {
        return 1;
    }
    if (!this.equals(predictorsEntropy)) {
        return 1;
    }
    return 0;
}

/**
 *
 * @return
 */
@Override
public String toString() {
    StringBuilder chain = new StringBuilder("Gen:");
    chain.append(gen).append("\n Predictors: ");
    for (Iterator<Integer> it = predictors.iterator(); it.hasNext();) {
        chain.append(it.next()).append(", ");
    }
    chain.delete(chain.length()-2,chain.length()-1).append("\n Logic:");
    if(TruthTable!=null)
    for (int i = 0; i < TruthTable.length; i++) {
        chain.append(Long.toBinaryString(TruthTable[i])).append(" ");
    }
    chain.append("\nI.M: ").append(mutualInformation);
    chain.append("\nGroup: ").append(groupingSelected);

    return chain.toString();
}

@Override
```

```
    public int hashCode() {
        int hash = 7;
        hash = 37 * hash + this.gen;
        hash = 37 * hash + Objects.hashCode(this.predictors);
        hash = 37 * hash + Arrays.hashCode(this.TruthTable);
        return hash;
    }

    @Override
    public boolean equals(Object obj) {
        if (obj == null) {
            return false;
        }
        if (getClass() != obj.getClass()) {
            return false;
        }
        final GenPredictors other = (GenPredictors) obj;
        if (this.gen != other.gen) {
            return false;
        }
        if (!Objects.equals(this.predictors, other.predictors)) {
            return false;
        }
        if (!Arrays.equals(this.TruthTable, other.TruthTable)) {
            return false;
        }
        return true;
    }
}
```

```
/*
 * CLASE InferenceNetwork
 */
package Inference;

import Resources.Resources;
import Networks.Network;
```

```
import Networks.MicroArray;
import java.io.*;
import java.util.*;

public class InferenceNetwork implements Serializable {

    /**
     * Arreglo que guarda los datos del microaaray
     */
    private MicroArray microArray;
    /**
     *
     */
    public static ArrayList<int[][]> combinationPredictors = new ArrayList();


    public enum ALGORITMOS_BUSCA {
        EXHAUSTIVE_I,
        SFS,
        SBS_E
    }

    public enum Penalizacao {
        WITHOUT_PENALTY,
        NOT_OBSERVED
    }

    public enum Tipo_Inferencia {
        WITHOUT_GROUPING,
        GROUPING_LINEAR,
        GROUPING_SELECTIVE,
        GROUPING_JOINT;

        String toSortString() {
            switch (this) {
                case WITHOUT_GROUPING: {
                    return "WG";
```

```
            }
            case GROUPING_LINEAR: {
                return "GL";
            }
            case GROUPING_SELECTIVE: {
                return "GS";
            }
            case GROUPING_JOINT: {
                return "GJ";
            }
            default:
                return null;
        }

    }
}

/**
 *
 * @param microArray
 */
public InferenceNetwork(MicroArray microArray) {
    this.microArray = microArray;
}

/**
 * @param microArray
 */
public void setMicroArray(MicroArray microArray) {
    this.microArray = microArray;
}

public MicroArray getMicroArray() {
    return microArray;

}
```

```
/**
 *
 * @param penalty
 * @param noGen
 * @param typeOfInference
 * @return
 */
private GenPredictors CalculatePredictorsExa(Penalizacao penalty,
        int noGen,
        Tipo_Inferencia typeOfInference) {
    GenPredictors bestPredict = new GenPredictors(noGen, new TreeSet(), penalty,
        microArray, typeOfInference);
    ArrayList<GenPredictors> listP = new ArrayList();
    double maxIM = 0;
    listP.add(bestPredict);
    int noPredict = 1;
    while (true) {
        //inicializa valores para n predictors
        ArrayList<GenPredictors> listBest = new ArrayList();
        double imActual = maxIM; //guardar info actual
        int[] combination = new int[noPredict];
        for (int i = 0; i < noPredict; i++) {
            combination[i] = i;
        }
        while (true) {
            //calcular informacion mutua
            TreeSet<Integer> pred = new TreeSet();
            for (int i = 0; i < combination.length; i++) {
                pred.add(combination[i]);
            }
            GenPredictors p = new GenPredictors(noGen, pred, penalty, microArray,
                typeOfInference);
            p.assignMutualInformation(p.createTable());
            //verificar si mejora la info actual
            if (p.mutualInformation > maxIM) {
                listBest.clear();
                listBest.add(p);
```

```
                maxIM = p.mutualInformation;//actualizar value de maxima
                    informacion mutua
            } else if (p.mutualInformation == maxIM && //misma informacion mutua
                    !listBest.isEmpty() && //algun predict
                    listBest.get(0).predictors.size() == p.predictors.size())//igual
                        numero de predictors {
            {
                listBest.add(p);
            }
            //ir al sigte conjunto de predictors
            if ((combination = Resources.NextSetPredictor(combination,
                microArray.getSizeGens())) == null) {
                break;//fin de predictors
            }
        }
        if (maxIM > imActual) {
            listP = listBest;
            noPredict++;
        } else {
            break;
        }
        bestPredict = listP.get((listP.size() > 1) ?
            Resources.random.nextInt(listP.size()) : 0);
    }
    return bestPredict;
}

/**
 *
 * @param penalty
 * @param noGen
 * @param typeOfInference
 * @return
 */
private GenPredictors CalculatePredictorsSFS(Penalizacao penalty,
        int noGen,
        Tipo_Inferencia typeOfInference) {
```

```
        double maxIM = 0;
        GenPredictors predict = new GenPredictors(noGen, new TreeSet(), penalty,
            microArray, typeOfInference);
        while (true) {
            //crear listas de almacenamiento de mejores predictors y de analizados
            ArrayList<GenPredictors> listBest = new ArrayList();
            // para cada gen
            for (int j = 0; j < microArray.getSizeGens(); j++) {
                TreeSet<Integer> pred = (TreeSet<Integer>) predict.predictors.clone();
                //tratar de agregar gen
                if (pred.add(j)) {
                    //si aun no fue analizado
                    GenPredictors p = new GenPredictors(noGen, pred, penalty,
                        microArray, typeOfInference);
                    p.assignMutualInformation(p.createTable());
                    if (p.mutualInformation > maxIM) {
                        listBest.clear();
                        maxIM = p.mutualInformation;
                        listBest.add(p);
                    } else if (p.mutualInformation == maxIM) {
                        if (!listBest.isEmpty()) {
                            listBest.add(p);
                        }
                    }
                }
            }
            if (!listBest.isEmpty()) {
                listBest = reduceByfrequency(listBest);
                predict = (listBest.size() == 1) ? listBest.get(0) :
                    listBest.get(Resources.random.nextInt(listBest.size()));
            } else {
                break;
            }
        }
        return predict;
    }
```

```
/**
 *
 * @param penalty
 * @param noGen
 * @param typeOfInference
 * @return
 */
private GenPredictors CalculatePredictorsSBS(Penalizacao penalty,
        int noGen,
        Tipo_Inferencia typeOfInference,
        int sizeInitial) {
    GenPredictors bestSBS;
    double maxIM = Double.NEGATIVE_INFINITY;
    //calcular melhor predict por busca exaustiva de tamano sizeInitial
    ArrayList<GenPredictors> listBest = new ArrayList();
    double imPresent; //guardar info actual
    int[] combination = new int[sizeInitial];
    for (int i = 0; i < sizeInitial; i++) {
        combination[i] = i;
    }
    while (true) {
        //calcular informacion mutua
        TreeSet<Integer> pred = new TreeSet();
        for (int i = 0; i < combination.length; i++) {
            pred.add(combination[i]);
        }
        GenPredictors p = new GenPredictors(noGen, pred, penalty, microArray,
            typeOfInference);
        p.assignMutualInformation(p.createTable());
        //verificar si mejora la info actual
        if (p.mutualInformation > maxIM) {
            listBest.clear();
            listBest.add(p);
            maxIM = p.mutualInformation;//actualizar value de maxima informacion
                mutua
        } else if (p.mutualInformation == maxIM) {
            listBest.add(p);
```

```
    }
    //ir al sigte conjunto de predictors
    if ((combination = Resources.NextSetPredictor(combination,
        microArray.getSizeGens())) == null) {
        break;//fin de predictors
    }
}
listBest = reduceByfrequency(listBest);
bestSBS = ((listBest.size() > 1)) ?
    listBest.get(Resources.random.nextInt(listBest.size()))
        : listBest.get(0);
//aplicar algoritmo sbs
int pr = 0;
combination = new int[sizeInitial];
for (Iterator<Integer> it = bestSBS.predictors.iterator(); it.hasNext(); pr++)
    {
    combination[pr] = it.next();
}
int noPred = sizeInitial;
while (noPred >= 1) {
    imPresent = maxIM;
    noPred--;
    listBest.clear();
    int i = 0;
    while (i < combination.length) {
        TreeSet<Integer> pred = new TreeSet();
        for (int j = 0; j < combination.length; j++) {
            if (i != j) {
                pred.add(combination[j]);
            }
        }
        i++;
        GenPredictors p = new GenPredictors(noGen, pred, penalty, microArray,
            typeOfInference);
        p.assignMutualInformation(p.createTable());
        //verificar si mejora la info actual
        if (p.mutualInformation > maxIM) {
```

```
                listBest.clear();
                listBest.add(p);
                maxIM = p.mutualInformation;//actualizar value de maxima
                    informacion mutua
            } else if (p.mutualInformation == maxIM) {
                listBest.add(p);
            }
        }
        if (imPresent <= maxIM) {
            listBest = reduceByfrequency(listBest);
            bestSBS = ((listBest.size() > 1)) ?
                listBest.get(Resources.random.nextInt(listBest.size()))
                    : listBest.get(0);
            if (noPred >= 1) {
                combination = new int[bestSBS.predictors.size()];
                pr = 0;
                for (Iterator<Integer> it = bestSBS.predictors.iterator();
                    it.hasNext(); pr++) {
                    combination[pr] = it.next();
                }
            }
        }
    }
    return bestSBS;
}

/**
 *
 * @param typeOfInference
 * @param penalty
 * @return
 */
private GenPredictors[]
    generateTemplateExhaustive(InferenceNetwork.Tipo_Inferencia typeOfInference,
        Penalizacao penalty) {
    int sizeGenes = microArray.getSizeGens();
    GenPredictors[] Template = new GenPredictors[sizeGenes];
```

```
    for (int iesimoGen = 0; iesimoGen < sizeGenes; iesimoGen++) {
        Template[iesimoGen] = CalculatePredictorsExa(penalty, iesimoGen,
            typeOfInference);
    }
    return Template;
}

/**
 *
 * @param typeOfInference
 * @param penalty
 * @return
 */
private GenPredictors[] generateTemplateSFS(InferenceNetwork.Tipo_Inferencia
    typeOfInference,
        Penalizacao penalty) {
    int sizeGens = microArray.getSizeGens();
    GenPredictors[] template = new GenPredictors[sizeGens];
    for (int iesimoGen = 0; iesimoGen < sizeGens; iesimoGen++) {
        template[iesimoGen] = CalculatePredictorsSFS(penalty, iesimoGen,
            typeOfInference);
    }
    return template;
}

/**
 *
 * @param typeOfInference
 * @param penalty
 * @return
 */
private GenPredictors[] generateTemplateSBS(InferenceNetwork.Tipo_Inferencia
    typeOfInference,
        Penalizacao penalty, int sizeInitial) {
    int sizeGens = microArray.getSizeGens();
    GenPredictors[] template = new GenPredictors[sizeGens];
    for (int iesimoGen = 0; iesimoGen < sizeGens; iesimoGen++) {
```

```
            template[iesimoGen] = CalculatePredictorsSBS(penalty, iesimoGen,
                typeOfInference, sizeInitial);
        }
        return template;
    }

    /**
     *
     * @param template
     * @param delta
     * @return
     */
    private Network CreateNetwork(GenPredictors[] template) {
        Network network = new Network(template.length,
            Network.Tipo_topologia.INFERRED);
        network.getPredictors().clear();
        for (int i = 0; i < template.length; i++) {

            if (template[i].predictors.size() > 0) {
                network.getPredictors().add(template[i].predictors);
            } else {
                network.getPredictors().add(new TreeSet());
            }
        }
        return network;
    }

    public Network inferNetwork(Tipo_Inferencia typeOfInference,
            ALGORITMOS_BUSCA search_algorithms,
            Penalizacao penalty, int size_top){
        GenPredictors[] template=null;
        switch(search_algorithms){
            case EXHAUSTIVE_I:template=generateTemplateExhaustive(typeOfInference,
                penalty);break;
            case SFS:template=generateTemplateSFS(typeOfInference, penalty);break;
            case SBS_E:template=generateTemplateSBS(typeOfInference, penalty,
                size_top);break;
```

```
        }
        return CreateNetwork(template);
    }
    private ArrayList<GenPredictors> reduceByfrequency(ArrayList<GenPredictors>
        predictors) {
        if (predictors.size() > 2) {
            ArrayList<GenPredictors> reduced = new ArrayList();
            int[] count = new int[microArray.getSizeGens()];
            for (int i = 0; i < predictors.size(); i++) {
                TreeSet<Integer> genPredictors = predictors.get(i).predictors;
                for (Iterator<Integer> it = genPredictors.iterator(); it.hasNext();) {
                    count[it.next()]++;
                }
            }
            int max = 0;
            for (int i = 0; i < predictors.size(); i++) {
                int value = 0;
                TreeSet<Integer> genPredictors = predictors.get(i).predictors;
                for (Iterator<Integer> it = genPredictors.iterator(); it.hasNext();) {
                    value += count[it.next()];
                }
                if (value > max) {
                    max = value;
                    reduced.clear();
                    reduced.add(predictors.get(i));
                } else if (value == max) {
                    reduced.add(predictors.get(i));
                }
            }
            return reduced;
        } else {
            return predictors;
        }
    }
}
```

```
/*
```

```
 * CLASE Networks
 */
package Networks;

import java.io.*;
import java.util.*;


public class MicroArray implements Serializable {

    private boolean[][] data;
    private int sizeGens;
    private int noExperiment;
    private long[][] state;
    private int[][] mapPrediction;

    public enum Separator {

        space(' '),
        semicolon(';');
        public char character;

        Separator(char c) {
            this.character = c;
        }

        public static Separator valueOf(char caracter) {
            for (Separator separador : EnumSet.allOf(Separator.class))
                if (separador.character == caracter) {
                    return separador;
                }
            return null;
        }

        @Override
        public String toString() {
            return Character.toString(character);
```

```
    }
}

public MicroArray() {
    this(0, 0);
}

public MicroArray(int noGens, int sizeInstanceOfTime) {
    this.sizeGens = noGens;
    this.noExperiment = sizeInstanceOfTime;
}

public boolean[][] getData() {
    return data;
}

public int getSizeGens() {
    return sizeGens;
}

public int getNroExperiments() {
    return noExperiment;
}

public void setSizeGens(int sizeGens) {
    this.sizeGens = sizeGens;
    ClearData();
}

public void setNoExperiments(int noExperiments) {
    this.noExperiment = noExperiments;
    ClearData();
}

public long[][] getState() {
    return state;
}
```

```
public int[][] getMapPrediction() {
    return mapPrediction;
}

public void Show() {
    for (int i = 0; i < sizeGens; i++) {
        for (int j = 0; j < noExperiment; j++)
            System.out.print(((data[i][j]) ? 1 : 0) + " ");
        System.out.println();
    }
}

public void ShowStates() {
    for (int i = 0; i < noExperiment; i++)
        System.out.println(Arrays.toString(state[i]) + " ");
}

public void BeginStates() {
    data = new boolean[sizeGens][noExperiment];
    mapPrediction = new int[sizeGens][noExperiment];
    state = new long[noExperiment][sizeGens / 64];
}

public void ClearData() {
    state = null;
    mapPrediction = null;
    data = null;
}

/**
 *
 * @param nameFile
 * @param separator
 * @throws FileNotFoundException
 */
public void SaveMicroArray(String nameFile, Separator separator) throws
```

```
        FileNotFoundException {
        if (!nameFile.endsWith(".may")) {
            nameFile += ".may";
        }
        SaveMicroArray(new File(nameFile), separator);
    }

    /**
     *
     * @param separate
     * @param fileNetwork
     * @throws FileNotFoundException
     */
    public void SaveMicroArray(File fileNetwork, Separator separate) throws
        FileNotFoundException {
        if (!fileNetwork.getName().endsWith(".may")) {
            fileNetwork = new File(fileNetwork.getAbsolutePath() + ".may");
        }
        try (PrintWriter file = new PrintWriter(fileNetwork)) {
            for (int j = 0; j < noExperiment; j++) {
                StringBuilder stLine = new StringBuilder();
                for (int i = 0; i < sizeGens; i++)
                    stLine.append((data[i][j]) ? "1" : "0").append(separate);
                file.println(stLine.toString());
            }
        }
    }

    public static MicroArray ReadMicroArray(String pathFile, Separator separator)
        throws FileNotFoundException, IOException {
        File arquivo = new File(pathFile);
        return MicroArray.ReadMicroArray(arquivo, separator);
    }

    public static MicroArray ReadMicroArray(File file, Separator separator) throws
        FileNotFoundException, IOException {
        //create object microarray
```

```
BufferedReader br = new BufferedReader(new FileReader(file));
// Lectura del fichero
//si separados es ; no se lee la primera linea
if (separator == Separator.semicolon)
    br.readLine();
//lista de state por instante de tiempo
ArrayList<boolean[]> liststates = new ArrayList();
ArrayList<long[]> listValuesStates = new ArrayList();
String line;
while ((line = br.readLine()) != null) {
    String[] ArrayStates = line.split(separator.toString());//arreglo de string
        de state
    boolean[] States = new boolean[ArrayStates.length];//arreglo booleano de
        estos guardo el iesimo estado del microarray
    long[] ValueState = new long[(ArrayStates.length % 64 == 0) ?
        ArrayStates.length / 64 : ArrayStates.length / 64 + 1];//guarda el
        valor del estado convertido a base diez
    for (int i = 0; i < ArrayStates.length; i++) {
        States[i] = ArrayStates[i].equals("1");
        if (States[i])
            ValueState[i / 64] += ((long) 1 << (i % 64));
    }
    liststates.add(States);
    listValuesStates.add(ValueState);
}
MicroArray microArray = new MicroArray(liststates.get(0).length,
    liststates.size());
microArray.BeginStates();
int nroEstado = 0;
for (Iterator<boolean[]> it = liststates.iterator(); it.hasNext();
    nroEstado++) {
    boolean[] estados = it.next();
    for (int i = 0; i < estados.length; i++)
        microArray.data[i][nroEstado] = estados[i];
}
nroEstado = 0;
for (Iterator<long[]> it = listValuesStates.iterator(); it.hasNext();
```

```
            nroEstado++)
            microArray.state[nroEstado] = it.next();
        return microArray;
    }

    public String chainOfStates(int col) {
        StringBuilder ce = new StringBuilder();
        int de = -1;
        long[] estado = state[col];
        for (int i = 0; i < state.length; i++) {
            if (Arrays.equals(estado, state[i])) {
                if (de == -1)
                    de = i;
                else {
                    ce.append("Of ").append(de).append(" a ").append(i).append("
                        Length:").append(i - de).append(" | ");
                    de = i;
                }
            }
        }
        if (ce.length() == 0)
            ce.append("Without Empty");
        return ce.toString();
    }

    public int[] valueOf(int i, int[] State) {
        int[] values = new int[State.length];
        for (int j = 0; j < State.length; j++)
            values[j] = (data[j][i]) ? 1 : 0;
        return values;
    }
}
```

Anexo C

Información Mutua

#	Predictores		Inform. Mutua
	X_1	X_2	
1	1402	3290	0.8777
2	3119	3290	0.8777
3	3290	4179	0.8777
4	3200	3290	0.8372
5	293	3290	0.8046
6	3290	4178	0.8046
7	1220	2471	0.7983
8	1481	3290	0.7913
9	1478	3290	0.7913
10	299	3290	0.7911

Cuadro C.1: gen 1879 represe. por i13056_1

#	Predictores		Inform. Mutua
	X_1	X_2	
1	4170	4572	0.6623
2	688	4290	0.6469
3	75	4170	0.6365
4	230	3242	0.6177
5	612	4144	0.6145
6	455	1903	0.6139
7	3664	4290	0.6003
8	728	2818	0.5988
9	540	4901	0.5987
10	455	1271	0.5953

Cuadro C.2: gen 4546 represe. por opff72413

#	Predictores		Inform. Mutua
	X_1	X_2	
1	1376	1509	0.8753
2	1300	3357	0.8668
3	52	4240	0.8621
4	1509	2182	0.8597
5	1216	1509	0.8518
6	75	2091	0.8242
7	2307	4240	0.8075
8	1220	4726	0.8009
9	3226	4240	0.7994
10	4240	4690	0.7994

Cuadro C.3: gen 3881 represe. por n132_136

#	Predictores		Inform. Mutua
	X_1	X_2	
1	965	1969	0.8400
2	1969	2214	0.8386
3	49	3808	0.8068
4	589	965	0.7939
5	589	2214	0.7924
6	3871	4637	0.7851
7	3767	4964	0.7681
8	737	4568	0.7646
9	1002	3871	0.7644
10	1039	3903	0.7588

Cuadro C.4: gen 4550 represe. por opff72425

#	Predictores		Inform. Mutua
	X_1	X_2	
1	3017	5033	0.5634
2	931	4235	0.5370
3	2180	5033	0.5364
4	761	1831	0.5323
5	455	1215	0.5305
6	4025	5033	0.5216
7	3826	4383	0.5198
8	1167	5033	0.5184
9	2546	3562	0.5174
10	97	969	0.5166

Cuadro C.5: gen 3206 represe. por m11919_1

#	Predictores		Inform. Mutua
	X_1	X_2	
1	323	940	0.7862
2	406	2262	0.7830
3	2262	4931	0.7830
4	2076	3352	0.7680
5	2262	4494	0.7517
6	2262	4887	0.7517
7	3865	5063	0.7345
8	1397	1969	0.7289
9	3339	3832	0.7277
10	460	940	0.7223

Cuadro C.6: gen 3907 represe. por n132_40

#	Predictores		Inform. Mutua
	X_1	X_2	
1	1339	4070	0.6795
2	3361	4099	0.6577
3	1339	4327	0.6526
4	2115	3694	0.6496
5	2617	3098	0.6397
6	1708	4070	0.6393
7	816	1821	0.6384
8	1339	1984	0.6341
9	406	3361	0.6341
10	3361	4931	0.6341

Cuadro C.7: gen 3625 represe. por m48835_1

#	Predictores		Inform. Mutua
	X_1	X_2	
1	1265	1684	0.7601
2	622	4070	0.7421
3	4070	4440	0.7421
4	4070	4463	0.7421
5	3290	4070	0.7244
6	1339	4070	0.7230
7	1984	3290	0.7223
8	249	3098	0.7170
9	622	4425	0.7170
10	4425	4440	0.7170

Cuadro C.8: gen 1949 represe. por i1689_2

#	Predictores		Inform. Mutua
	X_1	X_2	
1	1448	2332	0.7288
2	1376	1509	0.7269
3	1300	3357	0.7221
4	52	4240	0.7137
5	1509	2182	0.7113
6	1216	1509	0.7048
7	75	2091	0.7039
8	2149	4481	0.6993
9	678	3162	0.6993
10	3163	3719	0.6915

Cuadro C.9: gen 2286 represe. por j2896_1

#	Predictores		Inform. Mutua
	X_1	X_2	
1	4070	4856	0.8054
2	2521	2864	0.7972
3	2864	4803	0.7972
4	468	3185	0.7934
5	3339	4332	0.7725
6	267	2900	0.7713
7	2473	4856	0.7691
8	1653	4332	0.7656
9	1653	4078	0.7652
10	3098	3185	0.7642

Cuadro C.10: gen 2391 represe. por j53_48

Bibliografía

[A. de la Fuente et al., 2002] A. de la Fuente, A., Brazhnik, P., and Mendes, P. (2002). Linking the genes: inferring quantitative gene networks from microarray data. *Trends in Genetics*, 18(8):395–398. 32

[A. J. Hartemink and Young, 2000] A. J. Hartemink, D. K. Gifford, T. S. J. and Young, R. A. (2000). Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks. pages 422–433. 32

[A. Ribeiro and Kauffman, 2006] A. Ribeiro, R. Z. and Kauffman, S. A. (2006). A general modeling strategy for gene regulatory networks with stochastic dynamics. *Journal of Computational Biology*, 13(9):1630–1639. 32

[Albert, 2005] Albert, R. (2005). Scale-free networks in cell biology. *J Cell Sci*, 118(21):4947–4957. 28

[Albert and Othmer, 2003] Albert, R. and Othmer, H. G. (2003). The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in drosophila melanogaster. *Journal of Theoretical Biology*, 223(1):1–18. 34

[Barabási, 2009] Barabási, A.-L. (2009). Scale-free networks: A decade and beyond. *Science*, 325(5939):412–413. 28, 30, 31

[Barrera et al., 2002] Barrera, J., Cesar-Jr., R. M., Dantas, D. O., Jr., D. C. M., and Trepode, N. W. (2002). From microarray images to biological knowledge. In *Proceedings of the Second Brazilian Symposium on Mathematical and Computational Biology*, `http://www.biomat.org/sbbmc/index.html`. e-papers. 2

[Barrera et al., 2004] Barrera, J., Cesar-Jr, R. M., Martins-Jr, D. C., Vencio, R. Z. N., Merino, E. F., Yamamoto, M. M., Leonardi, F. G., Pereira, C. A. B., and del Portillo, H. A. (2004). A

new annotation tool for malaria based on inference of probabilistic genetic networks. In *Fifth International Conference for the Critical Assessment of Microarray Data Analysis (CAMDA)*, pages 36–40, Durham. (Winner of the Best Presentation Award). 70, 71, 75

[Barrera et al., 2007] Barrera, J., Cesar-Jr, R. M., Martins-Jr, D. C., Vencio, R. Z. N., Merino, E. F., Yamamoto, M. M., Leonardi, F. G., Pereira, C. A. B., and del Portillo, H. A. (2007). Constructing probabilistic genetic networks of *Plasmodium falciparum* from dynamical expression signals of the intraerythrocytic development cycle. In *Methods of Microarray Data Analysis V*, chapter 2, pages 11–26. Springer. 16, 34, 42, 43, 44, 45, 71, 73, 75

[Bozdech and DeRisi, 2005] Bozdech, Z., L. M. P. B. W. E. Z. J. and DeRisi, J. (2005). The transcriptome of the intraerythrocytic developmental cycle of plasmodium falciparum. *PLoS Biol*. 15, 70, 71, 75

[Butte and Kohane, 2000] Butte, A. and Kohane, I. (2000). Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. In *Pacific Symposium on Biocomputing*, pages 418–429. 45, 62

[Castillo, 2012] Castillo, M. S. (2012). Métodos bayesianos para la estimación de redes reguladoras de genes y de perfiles de protínas a partir de microarrays de expresión genética. Tesis doctoral, Universidad de Granada. 17, 19, 25, 33

[Costa et al., 2008] Costa, L. F., Rodrigues, F. A., and Cristino, A. S. (2008). Complex networks: the key to systems biology. *Genetics and Molecular Biology*, 31(3):591–601. 28

[Costa et al., 2007] Costa, L. F., Rodrigues, F. A., Travieso, G., and Villas-Boas, P. R. (2007). Characterization of complex networks: a survey of measurements. *Advances in Physics*, 56(1):167–242. 28, 30, 31

[Crick, 1970] Crick, F. (1970). Central dogma of molecular biology. *Nature*, 227(5258):561–563. 3

[Datta and Dougherty, 2009] Datta, A. and Dougherty, E. R. (2009). Introducction to genomic signal processing with control. *CRC Press*, pages 9–115. 16, 26

[Davidich and Bornholdt, 2008] Davidich, M. I. and Bornholdt, S. (2008). Boolean network model predicts cell cycle sequence of fission yeast. *PLoS ONE*, 3(2):e1672. 34

[de Salud, 2015] de Salud, M. (2015). Boletín epidemiológico. Website. 5

[de Salud Dirección General de Epidemiología, 2014] de Salud Dirección General de Epidemiología, M. (2014). "boletín epidemiológico 2014 se-21". "Website". 56

[DeRisi et al., 1997] DeRisi, J. L., Iyer, V. R., and Brown, P. O. (1997). Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science*, 278(5338):680–686. 25

[D'haeseleer et al., 1999] D'haeseleer, P., Liang, S., and Somgyi, R. (1999). Tutorial: Gene expression data analysis and modeling. In *Pacific Symposium on Biocomputing*, Hawaii. 37

[Dougherty, 2011] Dougherty, E. R. (2011). Validation of gene regulatory networks: scientific and inferential. *Briefings in Bioinformatics*, 12(3):245–252. 2, 36, 39, 40, 41, 42

[Dougherty et al., 2001] Dougherty, E. R., Barrera, J., Mozelle, G., Kim, S., and Brun, M. (2001). Multiresolution analysis for optimal binary filters. *J. Math. Imaging Vis.*, 14(1):53–72. 10

[Dougherty et al., 2007] Dougherty, E. R., Brun, M., Trent, J., and Bittner, M. L. (2007). A conditioning-based model of contextual regulation. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*. 55

[Dougherty et al., 2000] Dougherty, E. R., Kim, S., and Chen, Y. (2000). Coefficient of determination in nonlinear signal processing. *EURASIP Journal on Signal Processing*, 80(10):2219–2235. 51

[Dougherty et al., 2008] Dougherty, J., Tabus, I., and Astola, J. (2008). Inference of gene regulatory networks based on a universal minimum description length. *EURASIP Journal on Bioinformatics and Systems Biology*, 2008:1–11. 4, 45

[Duda et al., 2000] Duda, R. O., Hart, P. E., and Stork, D. (2000). *Pattern Classification*. Wiley-Interscience, NY. 51

[Erdös and Rényi, 1959] Erdös, P. and Rényi, A. (1959). On random graphs. *Publ. Math. Debrecen*, 6:290–297. 28, 29

[Espinosa-Soto et al., 2004] Espinosa-Soto, C., Padilla-Longoria, P., and Alvarez-Buylla, E. R. (2004). A gene regulatory network model for cell-fate determination during arabidopsis thaliana flower development that is robust and recovers experimental gene expression profiles. *Plant Cell*, 16(11):2923–2939. 34

[Faith et al., 2007] Faith, J., Hayete, B., Thaden, J., Mogno, I., Wierzbowski, J., Cottarel, G., Kasif, S., Collins, J., and Gardner, T. (2007). Large-scale mapping and validation of escherichia coli transcriptional regulation from a compendium of expression profiles. *PLoS Biology*, 5(1):259–265. 45

[Faure et al., 2006] Faure, A., Naldi, A., Chaouiya, C., and Thieffry, D. (2006). Dynamical analysis of a generic boolean model for the control of the mammalian cell cycle. *Bioinformatics*, 22(14):e124–131. 34

[Friedman et al., 2000] Friedman, N., Linial, M., Nachman, I., and Pe'er, D. (2000). Using bayesian networks to analyze expression data. *Journal of Computational Biology*, 7:601–620. 34, 39

[Hashimoto et al., 2004] Hashimoto, R. F., Kim, S., Shmulevich, I., Zhang, W., Bittner, M. L., and Dougherty, E. R. (2004). Growing genetic regulatory networks from seed genes. *Bioinformatics*, 20(8):1241–1247. 45

[Hecker et al., 2009] Hecker, M., Lambeck, S., Toepfer, S., van Someren, E., and Guthke, R. (2009). Gene regulatory network inference: data integration in dynamic models-a review. *Biosystems*, 96:86–103. 3

[Hogeweg, 2011] Hogeweg, P. (2011). *The roots of bioinformatics in theoretical biology*. PLoS computational biology. 1

[Hopkins, 2015] Hopkins, J. (2015). "life cycle of malaria parasite". "Website". IV, 57

[Hovatta, 2005] Hovatta, I Kimppa, K. L. (2005). *DNA microarray data analysis CSC, Ltd, 2nd Edition*. Scientic Computing. 39, 58

[Hsing et al., 2005] Hsing, T., Liu, L.-Y., Brun, M., and Dougherty, E. R. (2005). The coefficient of intrinsic dependence (feature selection using el cid). *Pattern Recognition*, 38(5):623–636. 55

[Huang et al., 2009] Huang, Y., Tienda-Luna, I. M., and Wang, Y. (2009). A survey of statistical models for reverse engineering gene regulatory networks. *IEEE signal processing magazine*, 26(1):76. 32

[Ivanov and Dougherty, 2006] Ivanov, I. and Dougherty, E. R. (2006). Modeling genetic regulatory networks: continuous or discrete? *Journal of Biological Systems*, 14(2):219–229. 35

[Jain et al., 2000] Jain, A. K., Duin, R. P. W., and Mao, J. (2000). Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37. 45, 46, 48

[Jong, 2002] Jong, H. D. (2002). Modeling and simulation of genetic regulatory systems: A literature review. *Journal of Computational Biology*, 9(1):67–103. 33, 34

[Karlebach and Shamir, 2008] Karlebach, G. and Shamir, R. (2008). Modelling and analysis of gene regulatory networks. *Nat Rev Mol Cell Biol*, 9(10):770–780. 34

[Kauffman, 1969] Kauffman, S. A. (1969). Homeostasis and differentiation in random genetic control networks. *Nature*, 224(215):177–178. 1, 28, 34, 37, 38

[Kauffman, 1993] Kauffman, S. A. (1993). *The Origins of Order*. Oxford University Press. 28

[Kelemen et al., 2008] Kelemen, A., Abraham, A., and Chen, Y. (2008). *Computational Intelligence in Bioinformatics*. Springer. 3, 34, 35, 39

[Kitano, 2002] Kitano, H. (2002). *Systems biology: a brief overview*. Science. 1

[Lähdesmäki et al., 2006] Lähdesmäki, H., Hautaniemi, S., Shmulevich, I., and Yli-Harjaa, O. (2006). Relationships between probabilistic boolean networks and dynamic bayesian networks as models of gene regulatory networks. *Signal Processing*, 86(4):814–834. 34

[Li et al., 2004] Li, F., Long, T., Lu, Y., Ouyang, Q., and Tang, C. (2004). The yeast cell-cycle network is robustly designed. *Proc. Natl. Acad. Sci. USA*, 101(14):4781–4786. 34, 38

[Li and Lu, 2005] Li, L. M. and Lu, H. H. S. (2005). Explore biological pathways from noisy array data by directed acyclic boolean networks. *Journal of Computational Biology*, 12(2):170–185. 34

[Liang et al., 1998] Liang, S., Fuhrman, S., and Somogyi, R. (1998). Reveal, a general reverse engineering algorithm for inference of genetic network architectures. In *Pacific Simposium on Biocomputing*, volume 3, pages 18–29. 45

[Lin, 1991] Lin, J. (1991). Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151. 51

[Lopes et al., 2011a] Lopes, F. M., Cesar-Jr, R. M., and Costa, L. F. (2011a). Gene expression complex networks: synthesis, identification and analysis. *Journal of Computational Biology*, 18:1353–1367. 37, 40, 44

[Lopes et al., 2010] Lopes, F. M., Martins-Jr, D. C., Barrera, J., and Cesar-Jr, R. M. (2010). SFFS-MR: a floating search strategy for grns inference. In *Pattern Recognition in Bioinformatics, Proceedings*, volume 6282 of *Lecture Notes in Computer Science*, pages 407–418, Nijmegen, Netherlands. Springer Berlin / Heidelberg. 52

[Lopes et al., 2014] Lopes, F. M., Martins-Jr, D. C., Barrera, J., and Cesar-Jr, R. M. (2014). A feature selection technique for inference of graphs from their known topological properties: revealing scale-free gene regulatory networks. *Information Sciences*. (in press). 44

[Lopes et al., 2008] Lopes, F. M., Martins-Jr, D. C., and Cesar-Jr, R. M. (2008). Feature selection environment for genomic applications. *BMC Bioinformatics*, 9(451). 52

[Lopes et al., 2009] Lopes, F. M., Martins-Jr, D. C., and Cesar-Jr, R. M. (2009). Comparative study of grn's inference methods based on feature selection by mutual information. In *IEEE International Workshop on Genomic Signal Processing and Statistics (GENSIPS)*, Minneapolis,MN,USA. 52

[Lopes et al., 2011b] Lopes, F. M., Oliveira, E. A., and Cesar-Jr, R. M. (2011b). Inference of gene regulatory networks from time series by tsallis entropy. *BMC Systems Biology*, 5:61. 44, 52

[M. et al., 2001] M., M. M., G., P., Vanderberg, Hafalla, J. C., Frevert, U., Nussenzweig, R. S., V., N., and A., R. (2001). Migration of plasmodium sporozoites through cells before infection. *Science*, 291(5501):141–4. 56

[M. López and Vega., 2002] M. López, P. and Vega., M. (2002). Microarrays y biochips de adn. technical report. *Fundación Genoma España*, pages 9–15. 24

[M. Sanchez-Castillo and Huang, 2011] M. Sanchez-Castillo, J. Meng, I. T.-L. and Huang, Y. (2011). Basisexpansion factor models for uncovering transcription factor regulatory networks. *Statistical Signal Processing Workshop (SSP)*. 32

[Mallat., 2009] Mallat., S. (2009). A wavelet tour of signal processing. *ELSEVIER*. 32

[Margolin et al., 2006] Margolin, A. A., Nemenman, I., Basso, K., Wiggins, C., Stolovitzky, G., Dalla-Favera, R., and Califano, A. (2006). Aracne: An algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics*, 7(Suppl 1). 45

[Martins-Jr et al., 2006] Martins-Jr, D. C., Cesar-Jr, R. M., and Barrera, J. (2006). W-operator window design by minimization of mean conditional entropy. *Pattern Analysis & Applications*, 9:139–153. 51

[Martins-Jr et al., 2010] Martins-Jr, D. C., Oliveira, E. A., Louzada, V. H., and Hashimoto, R. F. (2010). Inference of restricted stochastic boolean grn's by bayesian error and entropy based criteria. In *15th Iberoamerican Congress on Pattern Recognition (CIARP)*, volume 6419 of *Lecture Notes in Computer Science*, pages 144–152. Springer-Verlag. 52

[Martins Junior, 2009] Martins Junior, D. C. (2009). *Seleção de características e predição intrinsecamente multivariada em identificação de redes de regulação gênica*. PhD thesis, Universidade de Sao Paulo. iv, 3, 14, 46, 52, 53, 54, 58, 61

[Montoya, 2014] Montoya, F. (2014). Gene networks inference through linear grouping of variables. Master's thesis, Center for Mathematics Computation and Cognition - Federal University of ABC. iv, 5, 11, 47, 60, 65, 72, 75

[N. Guelzim and Képés, 2002] N. Guelzim, S. Bottani, P. B. and Képés, F. (2002). Topological and causal structure of the yeast transcriptional regulatory network. *Nature genetics*, 31(1):60. 28

[Nakariyakul and Casasent, 2009] Nakariyakul, S. and Casasent, D. P. (2009). An improvement on floating search algorithms for feature subset selection. *Pattern Recognition*, 42(9):1932–1940. 48

[Peng et al., 2005] Peng, H., Long, F., and Ding, C. (2005). Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. *IEEE TPAMI*, 27(8):1226–1238. 45

[Porter et al., 2001] Porter, D. A., Krop, I. E., Nasser, S., Sgroi, D., Kaelin, C. M., Marks, J. R., Riggins, G., and Polyak, K. (2001). A sage (serial analysis of gene expression) view of breast tumor progression. *Cancer Research*, 61(15):5697–5702. 45

[Pudil et al., 1994] Pudil, P., Novovičová, J., and Kittler, J. (1994). Floating search methods in feature selection. *Pattern recognition letters*, 15(11):1119–1125. 48, 49

[RB, 2006] RB, B. (2006). *Doing Your Dissertation in Business and Management: The Reality of Research and Writing*. Sage Publications. 7

[Reis,] Reis, M. d. S. *Minimização de funções decomponíveis em curvas em U definidas sobre cadeias de posets–algoritmos e aplicações.* PhD thesis, Universidade de São Paulo. IV, 49

[Ris et al., 2008] Ris, M., Martins-Jr, D. C., and Barrera, J. (2008). A branch-and-bound optimization algorithm for u-shaped cost functions on boolean lattices. *(submitted).* 48

[S. Das and Welch, 2008] S. Das, D. Caragea, W. H. H. and Welch, S. M. (2008). Computational methodologies in gene regulatory networks. *IGI Global.* 32

[Sánchez and Thieffry, 2001] Sánchez, L. and Thieffry, D. (2001). A logical analysis of the drosophila gap-gene system. *Journal of Theoretical Biology*, 211(2):115–141. 34

[Saunders and P, 2007] Saunders, M, L. and P, Thornhill, A. (2007). *Research Methods for Business Students, 4th edition.* Prentice Hall. 6

[Shalon et al., 1996] Shalon, D., Smith, S. J., and Brown, P. O. (1996). A dna microarray system for analyzing complex dna samples using two-color fluorescent probe hybridization. *Genome Res*, pages 639–45. 2

[Shmulevich and Dougherty, 2007] Shmulevich, I. and Dougherty, E. R. (2007). *Genomic Signal Processing.* Princeton University Press, New Jersey. 33, 35, 38

[Shmulevich et al., 2002] Shmulevich, I., Dougherty, E. R., Kim, S., and Zhang, W. (2002). Probabilistic boolean networks: a rule-based uncertainty model for gene regulatory networks. *Bioinformatics*, 18(2):261–274. 3, 34, 35

[Snoep and Westerhoff, 2005] Snoep, J. L. and Westerhoff, H. V. (2005). From isolation to integration, a systems biology approach for building the silicon cell. *Topics in Current Genetics*, 13:13–30. 2

[Somol and Pudil, 2004] Somol, P. and Pudil, P. (2004). Fast branch & bound algorithms for optimal feature selection. *Pattern Analysis and Machine Intelligence*, 26(7):900–912. 48

[Somol et al., 1999] Somol, P., Pudil, P., Novovičová, J., and Paclık, P. (1999). Adaptive floating search methods in feature selection. *Pattern recognition letters*, 20(11-13):1157–1163. 48

[Styczynski and Stephanopoulos, 2005] Styczynski, M. P. and Stephanopoulos, G. (2005). Overview of computational methods for the inference of gene regulatory networks. *Computers & Chemical Engineering*, 29(3):519–534. 35

[Theodoridis and Koutroumbas, 2006] Theodoridis, S. and Koutroumbas, K. (2006). *Pattern Recognition*. Elsevier, Academic Press, Amesterdam, New York. 45, 51

[Velculescu et al., 1995] Velculescu, V. E., Zhang, L., Vogelstein, B., and Kinzler, K. W. (1995). Serial analysis of gene expression. *Science*, 270:484–487. 2

[Waddington, 1972] Waddington (1972). International union of biological sciences e union internationale des sciences biologiques. *Towards a theoretical biology*. 1

[Wang et al., 2009] Wang, Z., Gerstein, M., and Snyder, M. (2009). Rna-seq: a revolutionary tool for transcriptomics. *Nat Rev Genet*, 10(1):57–63. 2

[Watts and Strogatz, 1998] Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of small-world networks. *Nature*, 393:440–442. 28, 29

[Zhang et al., 2006] Zhang, Y., Qian, M., Ouyang, Q., Deng, M., Li, F., and Tang, C. (2006). Stochastic model of yeast cell-cycle network. *Physica D*, 219(1):35–39. 34, 38

[Zhao et al., 2008] Zhao, W., Serpedin, E., and Dougherty, E. R. (2008). Inferring connectivity of genetic regulatory networks using information-theoretic criteria. *IEEE/ACM TCBB*, 5(2):262–274. 45