Machine Learning para la predicción de interacciones entre microARN y ARN mensajeros

Manuel Merino Monge
Máster universitario de Bioinformática y bioestadística (UOC-UB)
Área del trabajo final: *Machine Learning*

Albert Pla Planas
José Antonio Morán Moreno
Enero de 2018

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Machine Learning para la predicción de interacciones entre micro ARN y ARN mensajeros*
Nombre del autor:	*Manuel Merino Monge*
Nombre del consultor/a:	*Albert Pla Planas*
Nombre del PRA:	*José Antonio Morán Moreno*
Fecha de entrega (mm/aaaa):	01 / 2018
Titulación::	*Máster en bioinformática y Bioestadística*
Área del Trabajo Final:	*Machine Learning*
Idioma del trabajo:	*Español*
Palabras clave	*microARN, selección de atributos, clasificadores.*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados y conclusiones del trabajo.*

Un micro ARN es una pequeña secuencia de ADN que se une a un ARN mensajero y regula la actividad desempeñada por dicho gen, ya sea por degradación de la secuencia o inhibición de la actividad. Las características físicas y químicas que determinan los sitios de unión del micro ARN con el ARN mensajero no son totalmente comprendida en la actualidad, por lo que se han desarrollado multitud de sistemas enfocados a predecir posibles secuencias dianas de un micro ARN en un gen. El objetivo del presente proyecto ha sido desarrollar una herramienta que permita evaluar si un micro ARN actúa como regulador de la expresión de un gen. Para ello se ha empleado técnicas de selección de atributos y técnicas de machine learning (redes neuronales y máquinas de soporte vectorial) empleando como lenguaje de programación Java y como librería de código de referencia Weka. Los datos empleados han sido obtenidos de repositorios públicos. De éstos, se han obtenido un total de 18091 casos de regulación positiva, frente a 16711 casos de candidatos de interacciones negativas. El resultado que se ha obtenido es de 86% de correcta clasificación, valor que se puede considerar adecuado a la hora de determinar posibles interacciones entre las secuencias.

Abstract (in English, 250 words or less):

Micro RNA are small DNA sequence that manages gene expresion. A micro RNA matches in a specific binding site. The reason of this one is unknown. Several tools have been developed to predict these interactions. The goal of this project is developing a Java software based on machine learning and feature selection to classify two sequences (RNA and micro RNA) using Weka library. Data were extrated from public repositories. 18091 positive cases and 16711 negative cases were obtained. The result is a high accuracy of correct classification (86%). Several problems were presented during developing, so that number of features and classifiers were limited.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

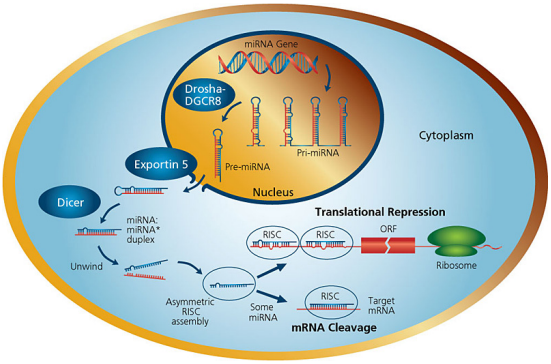
Introducción: bases biológicas de los micro ARN y sus aplicaciones

El ADN es un elemento microscópico que se encuentra en el interior de las células (en el núcleo o en el citoplasma dependiendo si se trata de células eucariotas o procariotas) que condensa toda la información necesaria que define y caracteriza el funcionamiento celular, y por consiguiente, el individuo al que pertenece. Éste está compuesto por estructuras denominadas cromosomas que contienen diferentes fragmentos de la información del ADN. En los seres humanos, en condiciones normales, cada cromosoma se encuentra duplicado formando 22 pares diferentes a los que hay que añadir un pareja más de cromosomas especiales: los que hacen referencia al sexo del individuo - *X* e *Y*. De este modo, existen 24 tipos de cromosomas distintos que se encuentran en el núcleo celular humano formando 23 parejas diferentes. Cada cromosoma está constituido por 2 cadenas simétricas que se retuercen y forma lo que se conoce como *estructura en doble hélice*, las cuales a su vez se componen de 4 tipos de moléculas diferentes llamadas *nucleótidos*: adenina (A), timina (T), citosina (C) y guanina (G).

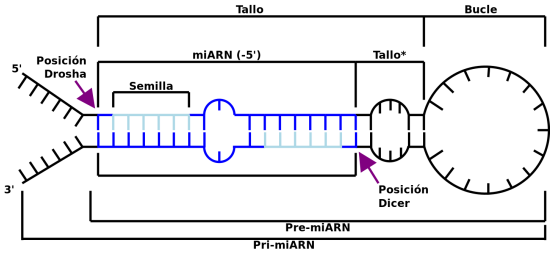
La actividad celular está principalmente regida por la información contenida en el ADN. Una de las más importantes es la síntesis de proteínas, las cuales se encuentran presentes en casi la totalidad de los procesos celulares (contracción muscular, transporte de oxígeno, receptores celulares, etc.). La síntesis se lleva a cabo por la acción conjunta de 2 tipos de ARN: ARN mensajero (ARNm) y ARN de transferencia (ARNt). El primero es una copia de un fragmento de un cromosoma que contiene lo que se denomina gen (unidad mínima de información que codifica la síntesis de una proteína u otro ARN) conformado por una única cadena donde el nucleótido T es sustituido por otro llamado uracilo (U), mientras el segundo es el responsable de ensamblar el producto funcional codificado en el gen que contiene el ARNm.

El estudio de este proceso de síntesis produjo que en 1993 se descubriera lo que se ha denominado **micro ARN (miARN)** [Lee et al., 1993]. Éstos son pequeños segmentos de ADN de entre 19 y 22 nucleótidos de doble cadena generados en el núcleo celular que intervienen en la regulación de la expresión genética, impidiendo la traducción de un ARNm, ya sea por degradación del ARNm (knock-out) o por inhibición de la traducción (knock-down).

La actividad de los miARN comienzan en el núcleo celular (figura 1.1a). La enzima *polimerasa II* cataliza la transcripción de un segmento de ADN en lo que se denomina *micro ARN primario (Pri-miARN)*. Esta partícula está compuesta por un dímero en



(a) Esquema de la síntesis de un microARN. Imagen obtenida de Wikipedia.



(b) Esquema estructural de un Pre-microARN [Jevsinek Skok et al., 2013].

Figura 1.1: Esquema de un microARN.

forma de 2 horquillas con unas prolongaciones en cadena simple, de modo que su tamaño impide que pueda viajar fuera del núcleo. La enzima *drosha* procede a eliminar estas prolongaciones separando las horquillas, lo que produce 2 *Pre-miARN* que sí pueden trasladarse fuera del núcleo (figura 1.1b). En el citoplasma, los Pre-miARN son procesados por otras enzimas llamadas *Dicer* que eliminan el bucle y segmenta el Pre-miARN en varios miARN dúplex de entre 19 y 22 pares de nucleótidos. Finalmente, el *complejo silenciador inducido por ARN (RISC)* divide el miARN en cadenas simples y se une a una de éstas. Posteriormente, el complejo RISC con el miARN se dirige a un ARNm y se adhiere a éste en la región donde es complementario el miARN, que se sitúa, por lo general, en la región no traducible 3' (3' UTR). De esta manera, se regula la expresión del gen asociado al ARNm mediante degradación o inhibición.

En una secuencia de ARNm existen multitud de subsecuencias complementarias en las que se podría unir un miARN. Sin embargo, en la mayoría de los casos esto no ocurre, sino que son en unas pocas localizaciones donde se produce dicha unión. La causa subyacente que hace que se rechace la mayoría de las subsecuencias, siendo válidas unas pocas, no son totalmente comprendidas en la actualidad. Este hecho ha producido el desarrollo de diversas herramientas bioinformáticas basadas en clasificadores enfocadas en predecir uniones entre miARN y ARNm [Peterson et al., 2014].

1.1. Interacciones miARN-ARNm: características principales

La unión de un miARN en una determinada región del ARNm que influye en la regulación de la expresión del gen se denomina *interacción* (figura 1.2). No obstante, esta complementariedad entre secuencias no es condición suficiente para la unión, sino que son necesarias otras características para que se produzca. Del acoplamiento entre secuencias se pueden extraer diferentes parámetros que se emplean en las herramientas de predicción de interacciones. A continuación detallaremos las más comunes.

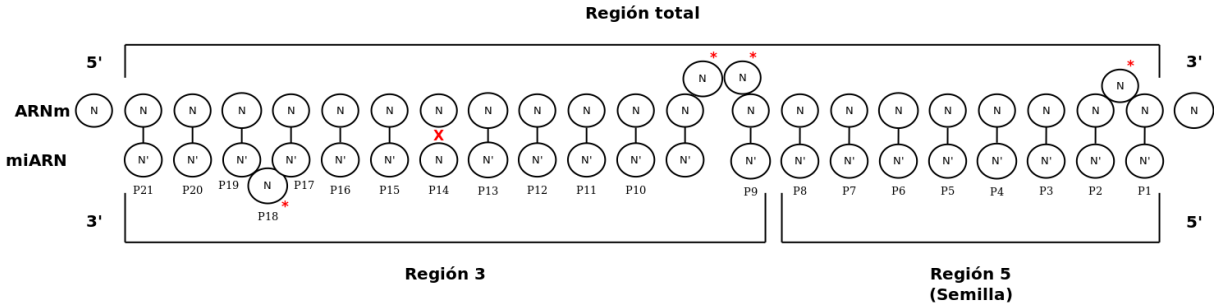


Figura 1.2: Ejemplo de estructura secundaria de pares miARN-ARNm [Yue et al., 2009]. *N* y *N'* hace referencia a un nucleótido y su complementario, **x** indica un incompatibilidad y * indica nucleótidos que han quedado desemparejados.

1.1.1. Región semilla

En un miARN maduro podemos diferenciar 2 áreas diferentes: la región 3' y la región 5' o semilla. Esta última se define como la secuencia de nucleótidos del miARN desde la posición 2 a la 8 (comenzando la enumeración desde el final 5' hacia el inicio del 3' del miARN) [Lee et al., 1993] (figura 1.2). En este área se pueden diferenciar 2 tipos de unión de nucleótidos: combinaciones perfectas o unión Watson-Crick (WC) (*A-U* y *G-C*) y uniones G-U (menos frecuentes).

Del estudio de la región semilla, se han identificado varios tipos de encajes de los cuales los más comunes son [Peterson et al., 2014, Yue et al., 2009]:

- *5mer-Z1*: desde la posición P1 a la P6, el tipo de la unión de los nucleótidos es WC o G-U, y al menos uno de ellos es de tipo G-U.
- *6mer*: 6 nucleótidos de la semilla presentan uniones de tipo WC.
- *6mer-GU*: desde la posición P2 a la P7, los nucleótidos encajan como WC o G-U, y al menos uno de ellos es de tipo G-U.
- *7mer-m8*: la unión de los nucleótidos es de tipo WC desde la posición P2 a la P8.
- *7mer-A1*: el nucleótido de la posición P1 del miARN es una A y desde la posición P2 a la P7 las uniones son de tipo WC.
- *8mer*: la región de semilla presenta uniones WC en todos ellos (P2-P8).

La región semilla es una de las características más determinante a la hora de localizar los lugares de unión de los miARN. No obstante, en los últimos años se ha descubierto que existen multitud de casos donde el ensamblaje de esta región no cumple ninguno de los tipos anteriores. Estos casos se denominan como sitio de encaje no canónica (SENC) [Kim et al., 2016], de los cuales podemos diferenciar 2 tipos [1] (figura 1.3):

- *Compensatoria*: se da en la región 3' del miARN. Secuencias complementarias fuera de la semilla.

[1] http://www.targetscan.org/docs/non-canonical.html

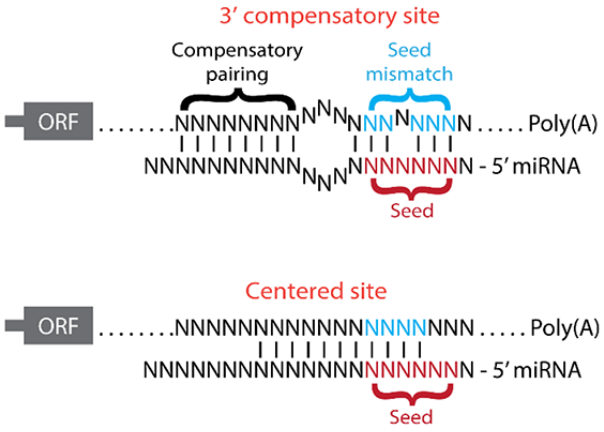


Figura 1.3: Tipos de sitios de uniones no canónicos. Imagen obtenida de www.targetscan.org.

- *Centrada*: casos en los que se carece de algunos de los tipos de encaje de semilla descrito previamente ni se ajusta al tipo compensatorio. Presenta entre 11 y 12 pares continuos de uniones de tipo WC desde la posición P4 a la P15.

1.1.2. Conservación

Hace referencia a la conservación de la secuencia del miARN entre diferentes especies, así como el grado de mantenimiento de la interacción. En general, se presupone que una mayor conservación de una secuencia de nucleótidos implica una mayor probabilidad de funcionalidad. De este modo, el nivel de conservación de la secuencia es mayor en la semilla que en el resto de la secuencia [Lewis et al., 2003], y en las interacciones puede darse que el grado de conservación al final de la región 3' del miARN compense incompatibilidades de semillas [Friedman et al., 2009]. Asimismo, el análisis de conservación puede evidenciar dianas de los miARN, en especial en las regiones 3' y 5' UTRs, lo que confiere a éstas cierto grado de funcionalidad de la que aparente carece.

1.1.3. Energía libre

Por lo general, la estabilidad de un sistema biológico puede medirse mediante la energía libre o energía de Gibbs (EG), de forma que cuanto menor sea su valor más estable se considera el sistema. En la mayoría de los casos, esto se traduce en que las dianas de un miARN suelen mostrar un EG más pequeño que aquellas que no lo son [Bonnet et al., 2004, Yue et al., 2009, Batuwita and Palade, 2009].

1.1.4. Accesibilidad de diana

La accesibilidad de diana representa el grado de apertura de la región 3' UTR al que se une el miARN, esto es, es una medida que indica el grado de facilidad con la que un miARN puede unirse con una subsecuencia candidata del ARNm [Kertesz et al., 2007].

1.1.5. Otras menos comunes

Las características anteriormente descritas son las más comúnmente usadas en las herramientas empleadas en predecir las interacciones. No obstante, existen otras características que se pueden encontrar en éstas:

- *Porcentajes de pares de nucleótidos en la interacción*: proporción de uniones A-U, G-C y G-U.
- *Número de protuberancias en el ARNm y en el miARN en la unión*: las protuberancias son nucleótidos libres de una unión que quedan desalineados de la secuencia (figura 1.2). El cómputo de estos tanto en el ARNm como en el miARN puede emplearse como característica a la hora de determinar los sitios de unión.
- *Número de incompatibilidades entre nucleótidos*: computo de nucleótidos que no encajan con su complementario, por ejemplo, A-C, C-C, G-A, etc.

1.2. Contexto y justificación del trabajo

Los mecanismos físico/químicos que condicionan que un miARN se acople a una secuencia complementaria y regule la expresión de un determinado gen y no a otro, no son totalmente comprendidos a día de hoy. Esto ha llevado a que se desarrolle multitud de herramientas informáticas focalizadas en predecir dianas entre miARN y ARNm, empleando para ello algoritmos de machine learning (ML) (en el capítulo 2 veremos los que hemos empleado en este trabajo de fin de máster (TFM)). Estos clasificadores precisan de un conjunto de atributos extraídos de las interacciones miARN-ARNm (IMAs) sobre los que operar, de forma que puedan predecir posibles puntos de unión. En la bibliografía podemos encontrar algunas de las siguientes herramientas [Peterson et al., 2014]:

1. *miRANDA-mirSVR*: herramienta que se basa en *miRanda* para identificar sitios de unión candidatos [Enright et al., 2003] y emplea un clasificador SVM para evaluarlo. *miRanda* no está especializada en ningún tipo de organismo en concreto y evalúa las siguientes características de los datos: comprueba las uniones WC, calcula la energía libre, de forma que las IMA con energía por debajo de un determinado umbral pasa a la siguiente fase donde el grado de conservación entre especies se emplea como filtro. El clasificador se desarrolló para humanos, ratas, ratones, moscas y gusanos, y los parámetros empleados son: tipo de encaje de semilla, energía libre, accesibilidad, contenido de A-U, posición del sitio de unión en la región 3' UTR, longitud de la región UTR, etc.
2. *TargetScan*: especializada en mamíferos, moscas y gusanos [Lewis et al., 2005]. Emplea el tipo de ensamblaje de semilla junto con el grado de conservación, pares compensatorios de la región 3' y contenido de A-U para generar un ranking de posibles dianas.
3. *DIANA-microT-CDS*: es uno de las primeras herramientas destinadas a predecir dianas en seres humanos [Maragkakis et al., 2009]. Emplea información sobre la región semilla, grado de conservación, accesibilidad, distancias entre sitios de uniones candidatos, contenido de A-U, la accesibilidad de la región 3' UTR, etc.

4. *MirTarget2*: emplea un clasificador SVM para evaluar un posible lugar de unión entre las secuencias. Los parámetros que se extraen de los candidatos son: tipo de ensamblaje de semilla, nivel de conservación, energía libre, accesibilidad, posición de la unión en la región 3' UTR, etc. Se puede utilizar para los siguientes organismos: humanos, ratones, ratas, perros y gallinas.

5. *rna22-GUI*: usa la energía libre y el tipo de encaje de semilla para predecir interacciones en humanos, ratones, moscas y gusanos [Loher and Rigoutsos, 2012].

6. *TargetMiner*: basado en un SVM para identificar potenciales lugares de unión sin centrarse en ningún organismos en concreto [Bandyopadhyay and Mitra, 2009]. Este clasificador fue entrenado con casos positivos extraídos de la base de datos *miRecord* [Xiao et al., 2009], y casos negativos obtenidos de un conjunto de datos de falsos positivos generados por otros algoritmos de predicción de dianas. Emplea características como: encaje de semilla, conservación, energía libre, accesibilidad, etc.

7. *SVMicr0*: la base de esta herramienta es un clasificador SVM [Liu et al., 2010] sin especificar ninguna especie. Las características extraídas de los datos son: encaje de semilla, conservación, energía libre, accesibilidad, proporción de incompatibilidades, uniones G-U, etc.

La capacidad de regulación del miARN podría ser una herramienta muy útil en terapias génicas para el tratamiento de diversas enfermedades, como el cáncer. Por ello, resulta de gran interés predecir las posibles dianas de los miARN, y así poder desarrollar tratamientos más eficaces, mejorar el conocimiento sobre la funcionalidad celular y determinar el papel que juega estas pequeñas moléculas en la actividad de la célula. Este potencial investigador, así como sus posibles aplicaciones medicina, es la principal razón por la que se ha seleccionado este tema para desarrollar el presente TFM. A su vez, *machine learning* es un área de investigación que ha permitido que la inteligencia artificial sea capaz de detectar emociones humanas, conducir vehículos de manera autónoma, etc. Es por ello, que la conjunción de ambas temáticas resulta de notable interés, y la razón de la selección de este tema como núcleo principal del proyecto final del máster. Asimismo, en el TFM se aplica los conocimientos adquiridos en la asignatura *Machine Learning* del máster en *Bionformática y Bioestadística* de la *Universitat Oberta de Catalunya* en que se adquirió la capacidad de identificar, analizar y sintetizar un problema de bioinformática definiendo su requerimientos, los conocimiento de las bases de datos biológicas públicas y cómo explotarlas.

1.3. Objetivos

El objetivo del proyecto es desarrollar una herramienta informática de predicción de dianas miARN que, a priori, desconocemos mediante la aplicación de técnicas ML y de selección de características (FS). Para la realización de esta tarea será necesario:

1. Evaluar diferentes técnicas de ML.

2. Estudiar y analizar bibliografía sobre IMAs para determinar qué parámetros extraer de los datos.

3. Conformar un conjunto de datos a partir de bases de datos públicas de dianas miARN que sirva para entrenar y evaluar los resultados de las técnicas ML.
4. Seleccionar los parámetros que influyen de forma significativa en el proceso de selección.
5. Desarrollar una aplicación con interfaz gráfica que facilite el uso de los clasificadores entrenados.

Durante el desarrollo del proyecto se usarán librerías existentes de ML y FS de libre distribución, ya que su correcto desempeño está garantizado al ser revisado el código por cientos de voluntarios. Respecto a los datos, se usarán base de datos pública que permitirá acelerar el proceso de análisis. Dado que las IMAs están presente en multitud de organismos, en este proyecto nos hemos centrándonos en *homo sapiens*. Finalmente, para asegurar que los parámetros que se extraen se adaptan correctamente al TFM, se va a desarrollar el código de programación que los calcule a partir de los datos en crudo. Al finalizar el presente TFM se espera haber desarrollado una aplicación que sea capaz de indicar si un determinado ARNm es diana de un miARN o no con un nivel de fiabilidad aceptable. Para ello se hará uso del lenguaje de programación Java y de la librería *Weka* que implementa multitud de técnicas ML y FS.

1.4. Metodología

Todo proyecto requiere del análisis detallado de qué se quiere conseguir y cómo se puede alcanzar. Por ello, se requiere un estudio minucioso de en qué estado está el tema que es objeto de estudio, determinar la información disponible para realizar la tarea, analizar qué técnicas se pueden aplicar y llevarlas a cabo. Estos pasos garantiza que, si se realiza cuidadosamente, se alcance con un alto grado de fiabilidad los objetivos marcados. Es por esto, que la mejor metodología a aplicar es la siguiente:

1. *Análisis del estado del arte actual en la detección de dianas*: se puede optar por buscar bibliografía en una biblioteca especializada, lo que garantiza un alto grado de fiabilidad de la información adquirida, pero es un proceso muy lento. Otra alternativa mucho más rápida sería realizar una búsqueda en Internet empleando algún buscador ya sea Google, Bing o Wikipedia. En cualquiera de los casos, la información que se puede obtener no siempre está validada por experto, con lo que se corre el riesgo de que no sea totalmente correcta o exacta. Una tercera alternativa es emplear un repositorio de artículos científicos. Esto garantiza que la información se obtiene rápidamente, además de veraz, al ser analizada por expertos
2. *Análisis de diferentes técnicas ML y FS, así como la eficacia en la clasificación en esta materia*: existen multitud de técnicas diferentes que se pueden emplear en este punto. El análisis de esta información se puede realizar estudiando de manera genérica los diferentes algoritmos, sin contrastar su uso en la bibliografía. En este enfoque se corre el riesgo de perder mucho tiempo en analizar técnicas que pueden haber sido descartadas por expertos por algún motivo. Otra alternativa que supera este riesgo es determinar qué técnicas son las más habituales y emplear esas mismas en el proyecto. Esta estrategia tiene el problema de que limita mucho la innovación y otras alternativas.

3. *Localizar muestras de casos positivos y negativos para usarlos en el desarrollo de los clasificadores*: se puede optar por realizar diferentes experimentos y verificar en qué casos se regula la actividad y en cuál no. Este enfoque permite una mayor fiabilidad de los datos, pero el incremento en tiempo y coste económico es notable. Otra alternativa es emplear muestras de otros estudios que están validadas y disponibles para el público.

4. *Extracción de características de un miARN para usarlos en las diferentes ML*: la estrategia que se puede emplear es similar al punto 2. Se puede optar por usar los atributos que otros expertos ya han empleado, lo que puede limitar la innovación y las posibilidades de mejorar los resultados previos, o se puede enfocar empleando los parámetros ya usados en otros estudios junto con nuevas características.

5. *Entrenamiento de diferentes ML y análisis de los resultados empleando las características seleccionadas por las técnicas FS*: en este punto se puede optar por implementar las diferentes técnicas desde cero, lo que se corre el riesgo de fallos en el código, u optar por librerías de código cuyo desempeño ha sido validado por expertos.

En el desarrollo del proyecto se opta por usar un repositorio de artículos científicos que asegura la fiabilidad de la información y a su vez obtener de éstos las muestras de datos positivos y negativos para el desarrollo de los clasificadores. Asimismo, de la bibliografía revisada se determinará que clasificador y técnicas de selección de atributos son las más empleadas. Esta información determinará qué técnicas usar, que se emplearán junto con alguna nueva que no se haya detectado para innovar en el análisis. Los parámetros a calcular de los datos se determinarán en base a los empleados en otros estudios relacionados con el tema. Finalmente, se opta por librerías existentes de ML y FS de libre distribución, ya que su correcto desempeño está garantizado al ser revisado el código por cientos de voluntarios, y para asegurar que los parámetros extraídos se adaptan correctamente al TFM, se desarrolla código que los calcule a partir de los datos en crudo.

1.5. Planificación

Para poder realizar con ciertas garantías el proyecto se define una serie de puntos a cumplir y su planificación temporal (figuras 1.4a y 1.4b). Las tareas son:

1. Estudiar diversas técnicas de clasificación de características: redes neuronales, SVM, etc.

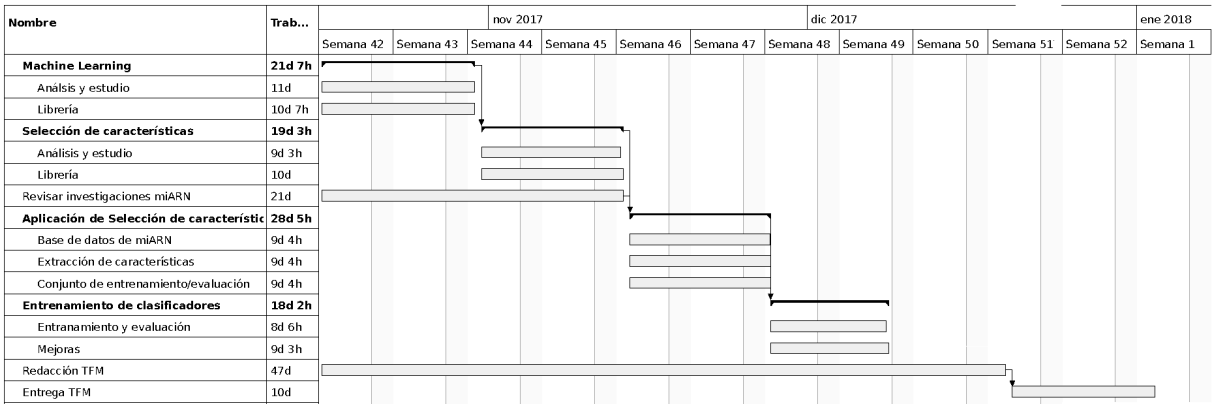
    a) Analizar técnicas de ML y su aplicabilidad en el proyecto.

    b) Buscar librerías que implementen técnicas de ML.

2. Estudiar técnicas de selección de características.

    a) Buscar librerías que implementen técnicas de selección de características.

3. Revisar el estado actual de las investigaciones en la detección de dianas de miARN.



(a) Diagrama de Grant: planificación temporal.

WBS	Nombre	Inicio	Fin	Trabajo	Duración
1	**Machine Learning**	**oct 16**	**oct 30**	**21d 7h**	**11d**
1.1	Análsis y estudio	oct 16	oct 30	11d	11d
1.2	Librería	oct 16	oct 30	10d 7h	10d 7h
2	**Selección de características**	**oct 31**	**nov 13**	**19d 3h**	**10d**
2.1	Análisis y estudio	oct 31	nov 13	9d 3h	9d 3h
2.2	Librería	oct 31	nov 13	10d	10d
3	Revisar investigaciones miARN	oct 16	nov 13	21d	21d
4	**Aplicación de Selección de características**	**nov 14**	**nov 27**	**28d 5h**	**9d 4h**
4.1	Base de datos de miARN	nov 14	nov 27	9d 4h	9d 4h
4.2	Extracción de características	nov 14	nov 27	9d 4h	9d 4h
4.3	Conjunto de entrenamiento/evaluación	nov 14	nov 27	9d 4h	9d 4h
5	**Entrenamiento de clasificadores**	**nov 27**	**dic 8**	**18d 2h**	**9d 3h**
5.1	Entranamiento y evaluación	nov 27	dic 8	8d 6h	8d 6h
5.2	Mejoras	nov 27	dic 8	9d 3h	9d 3h
6	Redacción TFM	oct 16	dic 19	47d	47d
7	Entrega TFM	dic 20	ene 2	10d	10d

(b) Tabla temporal de la planificación.

Figura 1.4: Planificación del trabajo fin de máster

4. Aplicación de técnicas de selección de atributos para seleccionar características de los miARN que se usarán en las ML.
   a) Buscar base de datos de miARN de la que extraer información y características.
   b) Extracción de características que se emplearán en las ML.
   c) Definición de los conjuntos de entrenamiento y evaluación de las ML.
5. Entrenamiento de los clasificadores.
   a) Entrenar y evaluar los clasificadores.
   b) Analizar posibles mejoras.
6. Redacción del documento de TFM.

Definida las tareas, los hitos de cada una de ellas serán:

1. Tarea 1: debe estar acaba para el 30 de octubre. Tras su finalización, se dispondrá de un informe que describirá qué técnicas de usará y por qué.
2. Tarea 2 y 3: finaliza el 13 de noviembre. Se generará un documento con la información obtenida.

3. Tarea 4: el día 27 de noviembre debe darse por concluida. Se redactará un informe con la procedencia de los datos, sus características y cómo se ha construido el conjunto de los datos.

4. Tarea 5: el 8 de diciembre debe darse por cerrada la tarea. Se realizará un resumen de los resultados obtenidos.

5. Tarea 6: la redacción del documento del TFM debe estar listo el día 19 de diciembre.

6. Tarea 7: entrega del TFM alrededor del día 2 de enero de 2018.

En las pruebas de evaluación (PEC) continuas 0 y 1 se definieron los objetivos y la planificación del TFM, de forma que el 2 y 10 de octubre de 2017 se estableció la estrategia que se iba a seguir en el proyecto. Para la fecha de entrega del PEC 2 de determinó que las tareas 1, 2 y 3 quedarán finalizada y la tarea 4 estuviera en desarrollo. Estos hitos fueron completados a tiempo, con la única excepción de seleccionar la librería de código a usar. Para la tarea 4 se determinó las fuentes de donde extraer la información, se estableció un conjunto inicial de atributos a calcular de los datos y la estrategia de entrenamiento y validación de los clasificadores fue definida, quedando por terminar la base de datos con la información de las interacciones. La finalización de las tareas 4 y 5 se planeó para el PEC 3. Estas se completaron a tiempo, de forma que la base de datos contenía toda la información necesaria para el proyecto, se determinaron las características a emplear en el entrenamiento de los clasificadores en base a los atributos propuestos por las técnicas FS. Se entrenaron 2 clasificadores en base a 3 conjuntos de parámetros y se desarrolló la interfaz Java para interactuar con el clasificador que mejor resultado arrojó. No obstante, diversos problemas en durante la tarea 4 produjeron un retraso considerable en la planificación (al menos 1 semana) lo que limitó el número parámetros a extraer de los datos, así como las técnicas ML y FS empleadas. Las tareas 6 y 7 se esperan finalizar para el PEC 4. La tarea 6 tiene un retraso de 1 semana, pero estará lista para la fecha de entrega del PEC 4.

1.6. Sumario de resultados

El proyecto que se ha desarrollado en el presente curso consiste en la obtención de un conjunto de muestras de datos (casos positivos y negativos) a partir de los cuales se extraen una serie de parámetros que se usan para entrenar los clasificadores. Con el objetivo de reducir la complejidad de los modelos y mejorar el rendimiento del proceso de entrenamiento se emplean técnicas de selección de características, de forma que se determina qué atributos son los más significativos. Finalmente, se analizaron técnicas de clasificación que determinarán el tipo de interacción que tienen un miARN con un determinado ARNm. De esta forma, se ha obtenido una base de datos con los casos positivos y negativos a usar en los clasificadores. Ésta fue rellenada con la información proveniente de 4 fuentes diferentes relacionadas con las secuencias completas de mis miARN, las secuencias completas de ARNm, conjunto de datos positivos proveniente de otro estudio, y un archivo con interacciones validadas. De la bibliografía se determinaron 26 parámetros que se extrajeron del conjunto de muestras (sobre 35000 casos) que se almacenaron en 2 archivos (uno de entrenamiento y otro de verificación de los clasificadores). Finalmente, los algoritmos FS generaron 3 conjunto de

parámetros diferentes que fueron empleados sobre 2 clasificadores: redes neuronales y máquinas de soporte vectorial (Support Vector Machine) (SVM). Finalmente, la red neuronal fue la que mejor resultado proporcionó, empleando sólo 5 atributos de los datos, por lo que se almacenó en un fichero que se emplea junto con la interfaz gráfica en Java para usar el clasificador para casos no contemplados.

Los ficheros de entrenamiento y verificación, la base de datos, el código Java y los datos de tarbase se adjuntará al proyecto como archivos auxiliares. Para los demás archivos empleados, se proporciona el enlace a la web desde donde se obtuvo.

1.7. Estructura de la memoria del TFM

Las diferentes tareas realizadas para conseguir alcanzar los objetivos del proyecto se detallan en el siguiente orden: en el capítulo 2 se describe las técnicas de clasificación empleadas en el trabajo, en el capítulo 3 se esbozan diferentes técnicas de selección de características que se emplearán como entrada de las técnicas de clasificación descritas en el capítulo 2, el capítulo 4 detalla todos los pasos llevados a cabo para generar los conjuntos de datos de entrenamiento y verificación para los algoritmos ML, el capítulo 5 describe los resultados obtenidos de la selección de características, los clasificadores y la estructura del programa Java en el que se ejecuta, y finalmente en el capítulo 6 se comentan los resultados contextualizándolo dentro de máster e indicando posibles mejoras.

Capítulo 2

Técnicas de machine learning

El término machine learning (ML) o clasificador identifica un conjunto de técnicas enfocadas en *aprender* a resolver un determinado problema de forma automática, entendiendo por *aprender* a la capacidad de identificar patrones en el conjunto de datos. Con esta información, se espera que el sistema no solo sea capaz de clasificar datos ya conocidos, sino que pueda predecir futuros comportamientos. Las técnicas ML requieren de un conjunto amplio de ejemplos, identificados como *conjunto de entrenamiento*, sobre los que basarse para poder determinar los patrones de comportamiento y predecir futuros valores. Uno de los principales problemas que puede detectarse en todo clasificador es el sobreajuste del modelo a los datos de entrenamiento, que limita su eficacia cuando los datos no están en el conjunto de entrenamiento. Esto suele aparecer cuando hay demasiados parámetros en el modelo respecto a la cantidad de datos disponibles. Los modelos con muchos atributos se ajustan a todos los datos, pero no generalizan tan bien como los modelos con pocos parámetros. Las técnicas de FS (capítulo 3) pueden solventar este problema. Por otro lado, el sobreajuste también puede provenir de un exceso de entrenamiento. El modelo, tras el proceso de entrenamiento, debe generalizar los resultados, de forma que pueda predecir el resultado de casos que no estaban contemplados en las muestras de entrenamiento. Sin embargo, un sobreentrenamiento puede generar que el algoritmo de aprendizaje se ajuste en exceso a unas características muy específicas de los datos dando como resultado que no generaliza el comportamiento global. Este sobreajuste se traduce en una disminución del error sobre los datos de entrenamiento, mientras que la actuación sobre muestras nuevas empeora.

Un tipo especial de clasificador es el que se puede denominar *ZeroR*. Éste realmente no es un clasificador, ya que no analiza los datos para determinar patrones, sino que su salida será siempre igual a la clase mayoritaria de los datos de entrada de entrenamiento, esto es, del total de muestras, se cuentan la proporción de cada clase, y de la que tenga mayor número de unidades será la empleada para clasificar todos los datos. Para el caso de interacciones miARN, si la clase mayoritaria en el conjunto de muestras de entrenamientos son *positiva*, entonces todos los datos de verificación serán evaluados como *positivos*, lo que generará un determinado nivel de precisión en la clasificación obtenida.

El proceso de generación de un clasificador se realiza en 2 fases: entrenamiento y verificación. En primera fase se suele tomar el 75 % de las muestras disponibles para que la técnica ML las analice y detecte los patrones de comportamientos. La segunda fase evalúa los resultados del clasificador sobre los datos restantes que no estaban en

el conjunto de entrenamiento, de forma que se verifica la eficacia que se obtuvo sobre las muestras del entrenamiento. Se espera que las técnicas ML muestren un resultado en la fase de verificación significativamente superior al clasificador *ZeroR*.

Existen multitud de técnicas diferentes de ML. En este TFM nos centraremos en 2 que detallaremos a continuación que son las que emplearemos en este proyecto: red neuronal (RN) y máquina de soporte vectorial SVM. El uso de redes neuronales para clasificar interacciones miARN no está muy extendido (de hecho, no he encontrado referencias al respecto), mientras que clasificadores SVM son ampliamente utilizados [Enright et al., 2003, Bandyopadhyay and Mitra, 2009, Liu et al., 2010]. De esta forma, con estas técnicas ML se busca innovar respecto a la bibliografía y comparar los resultados con una de las técnicas más usadas en miARN.

2.1. Redes neuronales

La base de funcionamiento en la que se inspira el clasificador conocido como redes neuronales es el mecanismo de activación de las neuronas: cada neurona es receptora de un conjunto de impulsos electrónicos, cuando el cúmulo de éstos sobrepasa un valor umbral, la neurona procede a propagar un impulso eléctrico hacia otra u otras neuronas con las que está conectada [Mitchell, 1997, Bash, 2015]. La siguiente ecuación simula este comportamiento:

$$y = g(\sum_{i=0}^{n} x_i \cdot w_{ji}) \tag{2.1}$$

donde *g(.)* es la función de activación que determina cuándo propagar el impulso y w_i pondera el peso de una determinada entrada a la neurona (variable x_i). De esta forma, las redes neuronales se conforman de nodos interconectados (figura 2.1).

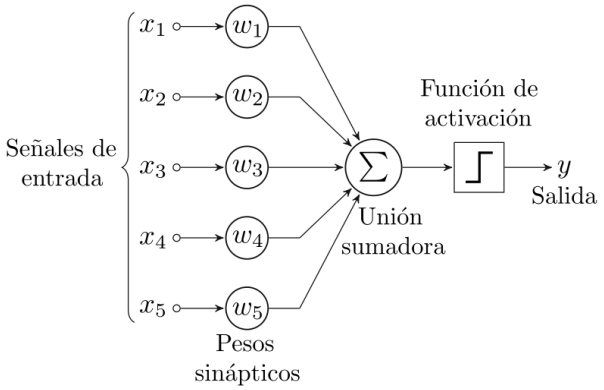


Figura 2.1: Modelo sencillo de una neurona. La salida de la unidad es $y = g(\sum_{i=0}^{n} x_i \cdot w_i)$ donde x_i es la salida de una neurona situada en la capa previa y w_i pondera el valor de dicha unidad en la unidad actual[Russell and Norvig, 2003]. Imagen obtenida de WikiCommons.

La función de activación *g(.)* es la responsable de eliminar, en la medida de lo posible, la ambigüedad en la salida de la red, de forma que se diferencie claramente los estados posibles que puede tomar. El rango de los datos de entrada y salida deben tenerse en consideración a la hora de determinar qué función de activación es la más adecuada para clasificar las muestras. Por ejemplo, para datos cuyos valores

están entre [0,$\pi/2$), la función tangente es adecuada, pero fuera de él no, ya que presenta una singularidad para $\pi/2$. Así pues, si el rango de salida debe estar acotado, entonces la función tangente tampoco es válida al no tener su salida acotada. En la literatura se pueden encontrar multitud de funciones de activación diferentes: función lineal, escalón, sigmoide, campana de Gauss, tangente hiperbólica, etc., siendo las más frecuentes las que son diferenciables al facilitar notablemente la tarea de aprendizaje/entrenamiento de los clasificadores.

2.1.1. Estructura de las redes neuronales

La redes neuronales están estructurada en capas, donde la salida de una neurona puede estar conectada con una o más neuronas de la capa siguiente (redes acíclicas o de alimentación hacia adelante), con neuronas de la capa que le precede, de su misma capa o incluso consigo mismo (redes recurrentes). En las redes acíclicas los datos se propagan en una única dirección desde la entrada hacia las salidas, sin que una neurona esté conectada con otras de la misma capa o de la capa anterior, de modo que estas redes *carecen de memoria*. Las redes recurrentes permiten una estructura mucho más compleja e interesantes pero difícil de analizar, al permitir que una neurona pueda estar conectada con cualquier neurona del sistema e incluso consigo misma, de modo que presenta una estructura con *memoria a corto plazo*.

La capas primera y última son conocidas como capas de *entrada* y *salida*, donde la primera no tiene una capa que le preceda, sino que sirve de entrada al sistema (por donde se recibe los datos), y la última no tiene capa que le siga, de manera que proporciona la salida de la red. El número de neuronas de ambas capas vendrán determinada por el problema a resolver: en la capa de entrada se tendrá tantas neuronas como número de elementos tenga el vector de datos de entrada, y el número de neuronas de la capa de salida vendrá determinado por la longitud del vector de salida que se desea.

2.1.2. Perceptrón

Una red neuronal con todas las entradas conectadas directamente a una única neurona de salida se llama perceptrón. Las redes acíclicas se pueden definir como un conjunto de perceptrón donde la salida de cada capa es la entrada del siguiente perceptrón.

El proceso de entrenamiento de las redes neuronales se basa en ajustar los pesos de las uniones neuronales para minimizar el error que se obtiene con los datos de entrenamiento. El método clásico empleado para valorar este error suele ser la diferencia al cuadrado de los datos con la salida esperada con respecto a los proporcionado por la red (ecuación 2.2).

$$E = \frac{1}{2}Err^2 = \frac{1}{2}(y - h_w(x))^2 \tag{2.2}$$

Minimizar el error cuadrático es la base del *método del descenso del gradiente*, que busca minimizar este error calculando las derivadas parciales de *E* con respecto a cada peso w_{ji}, de forma que se busca la configuración de los pesos que corresponda al mínimo global de la función de error. No obstante, por lo general este mínimo se desconoce, por lo no siempre es posible obtenerlo, y se buscan alternativas que

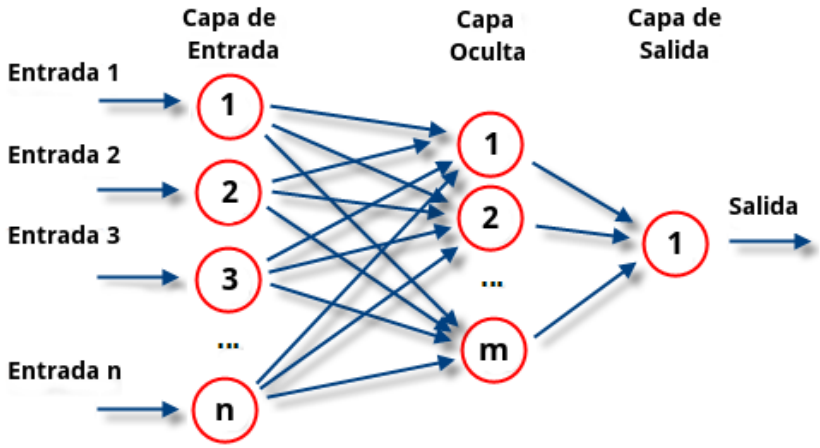


Figura 2.2: Red neuronal multicapa. Imagen obtenida de WikiCommons.

aproximen a este valor (mínimos locales). La idea general de la técnica se basa en calcular la dirección de máxima variación del error que viene dada por el gradiente del error en función de los pesos de la capa ($\nabla E(W)$). Con esta información se procede a actualizar los pesos siguiendo el sentido contrario del gradiente (máximo decrecimiento del error). De este modo, el proceso se va desplazando por la superficie del error hasta alcanzar un mínimo local. Matemáticamente, esto se expresa como sigue:

$$\frac{\partial E}{\partial W_j} = Err \cdot \frac{\partial E}{\partial W_j} = Err \cdot \frac{\partial}{\partial W_j} g\left(y - \sum_{j=0}^{n} W_j \cdot x_j\right) = -Err \cdot g'(in) \cdot x_j \tag{2.3}$$

donde $g'(.)$ es la derivada de la función de activación, y los pesos se actualizan en base a la ecuación 2.4:

$$W_j \leftarrow W_j + \alpha \cdot Err \cdot g'(in) \cdot x_j \tag{2.4}$$

donde α es la tasa de aprendizaje que evita comportamientos anómalos en este proceso.

2.1.3. Redes neuronales multicapas acíclicas

Las redes multicapas acíclicas están constituidas por perceptrones distribuidos en capas donde la salida de una es la entrada de la siguiente (figura 2.2). Las capas intermedias u ocultas permiten ampliar el espacio de datos que puede ser representado por la red, mejorando la adaptación y flexibilidad del modelo. El número de neuronas en cada de las capas ocultas no viene definido a priori, al no existir un procedimiento que determine los valores óptimos de éstas, de manera que es responsabilidad del investigador definir los valores.

Los algoritmo de entrenamiento de este tipo de redes se basan en el descrito para un perceptrón, con la principal diferencia que ahora tendremos un vector como salida de la red. En esta estructura el error de salida es igual que en un perceptrón ($y - h_w$), sin embargo el error de las capas intermedias es desconocido. Para superar este inconveniente, se propaga el error de salida hacia las capas ocultas basándonos en

que cada nodo oculto colabora en el error de salida. Este proceso de *propagación hacia atrás* proviene directamente del gradiente del error total.

La actualización de los pesos del vector de salida sigue la regla usada en un perceptrón adaptada a múltiples unidades de salida:

$$W_{j-1,i} \leftarrow W_{j,i} + \alpha \cdot Err_i \cdot g'(in) \cdot x_j = W_{j,i} + \alpha \cdot \Delta_i \cdot x_j \tag{2.5}$$

donde Err_i es el error de salida de la unidad i en la capa de salida.

Para actualizar los pesos de las conexiones entre las neuronas ocultas se ha de definir una cantidad análoga al término del error de los nodos de salida. La unidad oculta j de la capa $i-1$ es responsable de una fracción del error de salida Δ_i. Esta fracción se asigna en base al peso de la conexión entre el nodo oculto y el de salida, y se propaga hacia la capa previa oculta $i-1$ para obtener una estimación del error de esta capa Δ_{i-1} (ecuación 2.6).

$$\Delta_{i-1} = g'(in_j) \sum_j W_{j,i} \Delta_i \tag{2.6}$$

La regla que se aplica para actualizar los pesos entre las entradas y la capa oculta que le sigue es casi idéntica a la de la capa de salida:

$$W_{k,j} \leftarrow W_{k+1,j} + \alpha \cdot \Delta_j \cdot x_k \tag{2.7}$$

El proceso de propagación hacia atrás se puede resumir en:

1. Calcular los valores Δ para la capada de salida.
2. Desde la salida, repetir en cada capa intermedia hasta alcanzar la entrada:
   a) Propagar hacia la capa previa los valores de Δ de la capa oculta actual.
   b) Actualizar los pesos entre capas.

La convergencia de esta técnica es muy lenta y no siempre garantiza alcanzar el mínimo global de la función. Es por eso, que se han desarrollado multitud de técnicas que mejoran este proceso de entrenamiento, como por ejemplo *Levenberg-Marquardt* que incorpora la matriz Jacobiana a los cálculos. No obstante, esto escapa al objetivo del TFM por lo que no profundizaremos más en el tema.

Las redes neuronales han sido empleadas empleado con éxito en multitud de ocasiones, como por ejemplo, el reconocimiento de caracteres escritos [Singh et al., 2010], conducción de una silla de ruedas motorizada [Barea et al., 2002], determinar el estado emocional de un sujeto [Frenger, 1999], predicción económica [Bahrammirzaee, 2010], etc. De esta forma, queda patente las ventajas de aplicar esta técnica: 1) se pueden emplear en clasificación y en predicción numérica, 2) capacidad de modelar problemas complejos, y 3) no establece conjetura sobre los datos y la relación entre ellos. Sin embargo, tiene como contrapartida que: 1) carga computacional alta y fase de entrenamiento lenta, 2) tiende al sobreajuste de los datos de entrenamiento, y 3) es una estructura de *caja negra*, con lo que se dificulta la interpretación del modelo resultante.

2.2. Máquinas de soporte vectorial

Los algoritmos de máquinas de soporte vectorial (Support Vector Machine) (SVM) se basan en la generación de un conjunto de hiperplanos que maximicen la distancias de los datos a éstos (figura 2.3) pudiendo emplearse para clasificación o para regresión de datos [Mitchell, 1997, Bash, 2015]. El conjunto de puntos más cercanos al hiperplano se les llama *vector soporte*.

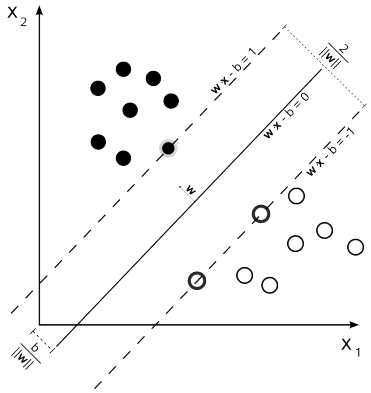


Figura 2.3: Idea base de los algoritmos SVM. Imagen obtenida de WikiCommons.

Centrándonos en el dominio $\Re^2$ para datos linealmente separables (existe un hiperplano que separa perfectamente los datos en 2 conjuntos de forma que los datos son todos correctamente clasificados), el objetivo es encontrar 2 hiperplanos que cumplan que

$$\overrightarrow{w} \cdot \overrightarrow{x} + b \geq +1 \tag{2.8}$$
$$\overrightarrow{w} \cdot \overrightarrow{x} + b \leq -1 \tag{2.9}$$

donde $\overrightarrow{w}$ es un vector de pesos, b es el sesgo de los datos y x son los datos. De este modo, la distancia entre ambos se define como $\frac{2}{||\overrightarrow{w}||}$ siendo el objetivo minimizar la distancia euclídea del vector $\overrightarrow{w}$ tal que cada dato es correctamente clasificado.

Si los datos no son linealmente separables, el objetivo será minimizar la siguiente expresión:

$$min\frac{1}{2}||\overrightarrow{w}||^2 + C\sum_{i=1}^{n}\epsilon_i \tag{2.10}$$

donde ϵ_i es el error y C compensa los errores de entrenamiento y los márgenes rígidos, creando así un margen blando que permita algunos errores en la clasificación a la vez que los penaliza. El parámetro C debe ser ajustado adecuadamente, ya que puede producir que un dato caiga en el lado equivocado del plano, y por tanto sea mal clasificado.

La relación entre las variables a clasificar no suele ser lineal por lo que hay que aplicar funciones que permitan una representación más compleja entre los datos. Aquí es donde aparece un aspecto fundamental de SVM: el *kernel*. Éste transforma los datos a una dimensión mayor con la esperanza de que se pueda generar un hiperplano que separe los datos que no son linealmente separables. La forma más simple de

calcularlas es mediante funciones lineales (rectas, planos, etc.) pero esto limita considerablemente la utilidad, ya que difícilmente los datos son linealmente separables. Para superar esta limitación se puede emplear *kernels* no lineales como polinomios, función sigmoide, campana de Gauss, etc. Estos proyectan la información a un espacio de características de mayor dimensión el cual aumenta la capacidad computacional de las máquinas de aprendizaje lineal.

Esta técnica ha sido empleada con éxito en el reconocimiento de comando de voz [Pereira et al., 2007], movimientos basados en electromiografía [Álvaro Ángel Orozco Gutiérrez, et a predicción de solubilidad de fármacos [Cano et al., 2017], predicción numérica en entornos financiero [Cao and Tay, 2003], etc. De esta forma, la técnica muestra ventajas muy importante frente a otras: 1) se puede emplear para clasificación y para predicción numérica (regresión), 2) no tiene al sobreajuste y es robusto al ruido de los datos, 3) es más fácil de usar que las redes neuronales, y 4) alta eficacia en la clasificación. Sin embargo, tiene como contrapartida que: 1) para encontrar la mejor *SVM* se necesita probar con varios kernels y parámetros, 2) el entrenamiento es lento, y 3) es un modelo de caja negra difícil de interpretar.

2.3. Evaluación de un clasificador

La evaluación de los algoritmos ML suelen realizarse en base a la *matriz de confusión*. Ésta es una estructura en forma de tabla que resume el desempeño de un algoritmo de aprendizaje supervisado. Las columnas de la matriz representan la etiqueta asignada por el clasificador, mientras las filas son los valores reales que debería tener una muestra. Con esto se consigue que una forma visual se pueda percibir el grado de confusión entre clases del sistema. Para el caso de una matriz de 2 clases binarias se puede extraer varios parámetros de interés: verdadero positivo (TP) informa del numero muestras positivas que son correctamente clasificadas, verdadero negativo (TN) identifica la cantidad de muestras negativas que han sido correctamente clasificadas, falso positivo (FP) cuenta el número de instancias que han sido incorrectamente clasificadas como positiva, y falso negativo (FN) mide cuántos casos son etiquetados como negativos cuando en realidad deberían ser positivos. En tabla 2.1 se muestra un esquema de una matriz de confusión de 2 clases.

Tabla 2.1: Matriz de confusión de 2 clases. Las columnas son la predicción del clasificador y las filas las clases reales.

	Positivo	Negativo
Positivo*	TP	FN
Negativo*	FP	TN

En base a la información contenida en la matriz de confusión se puede calcular los siguientes parámetros que miden la efectividad del clasificador: *exactitud (Pre)* que evalúa la eficacia general del clasificador (ecuación 2.11), *sensibilidad (Se)* que mide la capacidad de detectar correctamente las clases (ecuación 2.12) y *Especifidad (Sp)* que calcula la probabilidad de que la etiqueta pertenezca a otra clase (ecuación 2.13). Estos valores facilita la interpretación de los resultados de lo clasificadores, de forma que a mayor valor de los 3 parámetros, mejor será el clasificador.

$$Pre = 100 \cdot (TP + TN) \Big/ (TP + TN + FP + FN) \tag{2.11}$$

$$Se = 100 \cdot TP \Big/ (TP + FN) \tag{2.12}$$

$$Sp = 100 \cdot TN \Big/ (TN + FP) \tag{2.13}$$

Adicionalmente a las características anteriores, otra técnica muy habitual que se emplea en la evaluación de las técnicas ML es la *validación cruzada*. Ésta consiste en entrenar diversas instancias del mismo clasificador sobre muestras de datos entrenamiento y verificación diferente (seleccionadas aleatoriamente), de forma que el resultado final de la fase de evaluación será el promedio de cada uno (figura 2.4). La generación de los grupos se conoce como *k-folding* donde *k* hace referencia al número total de grupos generados. El objetivo es obtener un valor más próximo al comportamiento real, minimizando el efecto de la partición de los datos y los valores iniciales de las técnicas ML, por ejemplo, el peso de las conexiones en las redes neuronales.

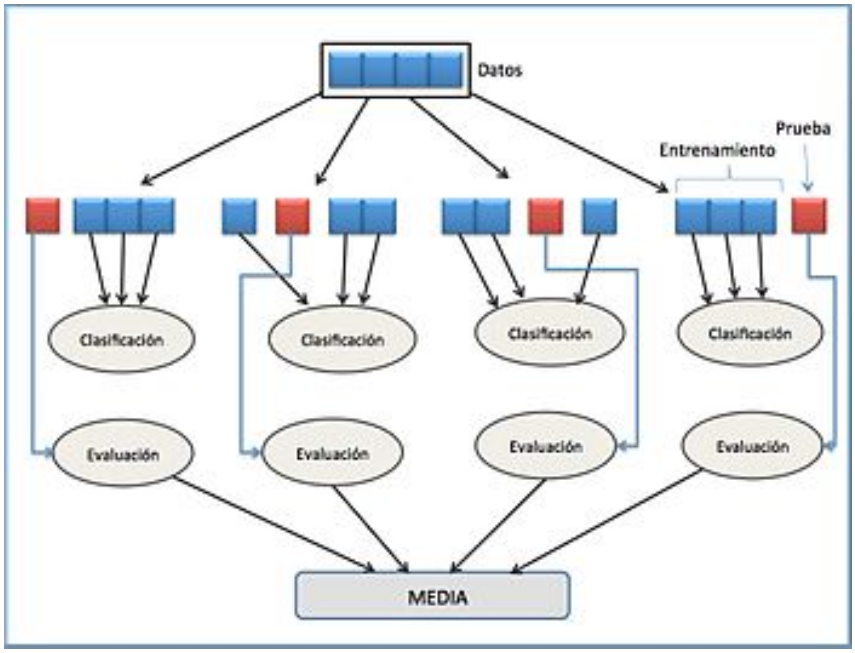


Figura 2.4: Esquema de validación cruzada de 4-folding. Imagen obtenida de Wikipedia.

Capítulo 3

Algoritmos de selección de características

La realización de experimentos para evaluar una determinada característica y establecer los principios básicos que rigen su comportamiento suelen tener asociados gran cantidad de parámetros. La relevancia de la información que aportan puede variar en función de si permiten o no distinguir los diferentes estados en los que puede estar el objeto de estudio. No obstante, cuando el número de variables extraídas en el experimento es grande, esta evaluación no es nada sencilla, y se debe recurrir a técnicas que permita distinguir la información relevante, puesto que es común que existan atributos irrelevantes y redundantes. Los algoritmos que permiten descartar estos parámetros nocivos se conocen con el nombre de *selección de características (Feature Selection) (FS)* [Chandrashekar and Sahin, 2014, Guyon et al., 2006], las cuales buscan facilitar la "visualización" de los datos, reducir el conjunto de datos y el almacenamiento requerido, mejorar la comprensión de los fenómenos que están detrás de los datos, incrementar la fiabilidad y el rendimiento de los sistemas que buscan predecir o clasificar los valores futuros del elemento de estudio.

El objetivo perseguido con estas técnicas es obtener un subconjunto de los atributos iniciales que optimice el modelo que se quiere desarrollar, ya que la interpretación del modelo obtenido resulta más sencilla cuanto menos atributos se usen. Para ello, datos redundantes, irrelevantes y la dimensionalidad de los mismos son elementos que pueden mermar notablemente el resultado. Los parámetros redundantes perjudicar y ralentizan el proceso de aprendizaje de ciertos modelos (SVM, árbol C4.5, Naive Bayes, etc.). Los atributos irrelevantes reducen la fiabilidad de los resultados y hacen confuso los modelos resultantes. La dimensionalidad está relacionada con el número de parámetros que se obtiene por cada una de las dimensiones del problema, lo que puede provocar que los datos crezcan exponencialmente y conducir a un sobreaprendizaje que reduce su fiabilidad en casos no contemplados en los datos de entrenamiento. Una consideración relevante y que hay que tener en cuenta a la hora de aplicar estos algoritmos es que ciertos atributos pueden resultar irrelevantes de forma individual, pero al actuar de forma conjunta pueden mejorar la separabilidad de los datos, y por tanto, incrementar la tasa de acierto de la clasificación. Así pues, las técnicas FS presentan las siguientes ventajas:

- Reducir el efecto de la dimensionalidad de los datos.
- Mejorar la capacidad de generalización del modelo al eliminar datos redundantes

e irrelevantes.

- Acelerar el proceso de aprendizaje del modelo.
- Reducir la complejidad y facilitar la interpretación del modelo obtenido.

3.1. Técnicas de selección de atributos

Las técnicas FS buscan un subconjunto mínimo de atributos que optimice la predicción/clasificación de los datos [Guyon and Elisseeff, 2003]. En una situación ideal, esto se realizaría combinando las n variables extraídas de todas las formas posibles (*búsqueda exhaustiva*), lo que genera 2^n posibles subconjuntos. Supongamos que se han medido 100 atributos diferentes en un experimento, esto genera $1{,}267651 \cdot 10^{+30}$, revelando que en la práctica este método resulta poco práctico cuando los atributos son relativamente grande. Por ello, es necesario buscar alternativas que garantice un equilibrio entre la relevancia de los datos seleccionado y el tiempo/coste empleado para obtenerlos.

La aplicación de técnicas FS requieren preprocesar los datos para mejorar el rendimiento y la fiabilidad de los resultados, consistente en: 1) análisis de los datos para eliminación de outliers y muestreo aleatorio simple para la selección de un conjunto de datos representativo más reducido, y 2) análisis de los atributos para su normalización, transformar parámetros discretos en variables *dummy* [1] y aplicar una estrategia a seguir con los datos faltantes. Asimismo, es necesario definir los 3 elementos centrales: estrategia de búsqueda, evaluación y verificación de los resultados:

- *estrategia de búsqueda*: cómo se va a ir seleccionando los parámetros para su posterior evaluación (búsqueda hacia adelante/atrás, mejor el primero, búsqueda genética,etc.).
- *evaluación de la relevancia*: calcular si se mejora los resultados sobre un conjunto de entrenamiento.
- *métodos de verificación*: validar que los atributos seleccionados mejoran el rendimiento general del sistema cuando se somete a datos que no están en el conjunto de entrenamiento.

Los algoritmos de selección de características se pueden agrupar en 2 clases:

- *métodos de ranking (ranking methods) (RM)*: evalúan la relevancia de cada variable de forma individual.
- *métodos de selección de subconjuntos (subset selection methods) (SSM)*: buscan un subconjunto de atributos más relevantes.

[1] https://en.wikipedia.org/wiki/Dummy_variable_(statistics)

3.2. Relevancia individual: métodos de ranking

Evalúan la relevancia de cada variable de forma individual, lo que hace que sean rápidos, eficientes y carezcan del problema de sobreajuste. El criterio subyacente se basa en la idea de que las variables con una mayor correlación con los datos garantizan predecir la clase con cierto grado de fiabilidad [Shardlow, 2016]. Los criterios que se pueden emplear son muchos, como por ejemplo el valor absoluto del coeficiente de Pearson, entropía, información mutua, etc. Establecido el criterio, se seleccionará los *k* atributos con mayor correlación con los datos.

Realizar evaluaciones monovariables presentan 2 grandes limitaciones: 1) pueden descartar características que individualmente son insignificantes, pero que junto a otras puede resultar relevantes al mejorar la separabilidad de los datos; y 2) pueden seleccionar variables que al contextualizarla con otras resultan poco útil debido a posible redundancia.

3.2.1. Correlación de Pearson

Método clásico basado en el coeficiente de correlación de Pearson (ecuación 3.1). La idea es que cuanto más próximo a 1 en términos absoluto, más relevante es la información que contiene la variable.

$$\rho(x,y) = \left| \frac{covarianza(x,y)}{varianza(x) \cdot varianza(y)} \right| \tag{3.1}$$

3.2.2. Entropía

La base de este método de selección está en la entropía de la información o de Shannon (ecuación 3.2), donde las variables con menor probabilidad son los que más información aportan. De esta forma, a menor valor de entropía más relevante se vuelve el atributo.

$$H(x) = -\sum_{i=1}^{k} p_{c_i} \cdot log_2(p_{c_i}) : p_{c_i} \text{ probabilidad del estado } c_i. \tag{3.2}$$

3.2.3. Información mutua

Mide la dependencia entre una variable y los posibles estados en los que puede encontrarse el objeto de estudio (ecuación 3.3), de forma que cuanto más próximo a 0 menor relevante es la información que aporta el atributo.

$$\begin{array}{l} l(x,y) = \sum_i \sum_c p(x=x_i, y=c) log\left(\frac{p(x=x_i,y=c)}{p(x=x_i)p(y=c)}\right) \\ p(x=x_i, y=c) = p(x=x_i)p(y=i) \text{ si x e y independientes.} \end{array} \tag{3.3}$$

3.3. Relevancia conjunta: selección de subconjuntos

La técnicas descritas en la sección 3.2 evalúan la usabilidad de los parámetros de forma individual sin tener en consideración el desempeño conjunto de los atributos [Guyon et al., 2006, Shardlow, 2016]. Por el contrario, la selección de subconjuntos evalúa grupos de parámetros completos para determinar cuáles son los más prometedores. A diferencia de los métodos de ranking que tienen implícito los métodos de búsqueda y evaluación de los atributos, es necesario definir cómo medir la bondad del subconjunto de los atributos y qué estrategia se va a emplear para determinar qué atributo incorporar al subconjunto.

La evaluación se puede realizar mediante selección de características correlacionadas (Correlation Feature Selection) (CFS) en el que se evalúa la similaridad de cada atributo con la clase de los datos y la correlación por redundancia entre los atributos (ecuación 3.4). Este método tiene la característica de ser rápido pero se corre el riesgo de eliminar atributos que no están correlacionados con la clase pero si con otros atributos.

$$Evaluacin(A_i) = \frac{\text{correlación con la clase}}{\text{correlación entre atributos}} \tag{3.4}$$

Otra alternativa a CFS son los métodos *Wrapper* que aplica algún algoritmo de clasificación y evalúa el porcentaje de acierto con el conjunto de atributos. Estos métodos proporcionan subconjuntos adecuados para un determinado clasificador trabajando con subconjuntos reales de los parámetros, sin embargo es muy lento y se tiene el riesgo de sobreaprendizaje.

Las estrategias de búsqueda que se pueden aplicar son las siguientes:

- *Mejor el primero*: se evalúa el efecto de añadir un atributo al subconjunto de parámetros. El que mejor resultado proporcione se incorporará al conjunto y se vuelve a repetir el proceso.
- *Búsqueda exhaustiva*: se analiza todas las posible combinaciones de los atributos. Esto hace que resulte muy lento.
- *Algoritmo genético*: se basa en la selección/sustitución de atributos de forma aleatoria. Se trata de un método rápido.
- *Greedy Step Wise*: se trata de ir modificando el subconjunto de atributos analizando el efecto de cada parámetro. Es un método muy rápido y existen dos estrategias diferentes:
  - *Hacia adelante*: partiendo de un conjunto vacío, se va añadiendo un atributo en cada iteración de forma que se selecciona el que mejor resultado proporcione.
  - *Hacia atrás*:partiendo del conjunto completo con todos los atributos, se va eliminando un atributo en cada iteración de forma que se elimina el que peor resultado da.
- *Búsqueda por ranking*: se ordena los atributos y se construye el subconjunto de forma incremental desde el mejor al peor, finalizando cuando la selección de nuevos atributos no mejore los resultados. Es un método de búsqueda rápido.

3.4. Algoritmo RELIEF

La técnica RELIEF es un proceso de selección individual basado en ranking (sección 3.2) pero que tiene ventajas de selección de subconjuntos y *Wrapper* (sección 3.3), siendo un algoritmo muy rápido, capaz de detectar atributos relevantes y que trabajan bien de forma conjunta, pero que no rechaza atributos redundantes [Guyon et al., 2006, Shardlow, 2016].

El proceso consiste en seleccionar aleatoriamente una muestra de los datos y buscar el vecino más próximo de la misma clase de la muestra (*hit*) y el más cercano de otra clase (*miss*). Con esto, se incrementa el peso de aquellos atributos que tiene el mismo valor para el dato *hit* y distinto para el dato *miss* (ecuación 3.5). El proceso se repite muchas veces hasta que la influencia de los atributos es ponderada por los pesos asociado, de forma que a mayor peso, más relevante resulta.

$$W_{i+1} = W_i - (X_i - hit_i)^2 + (X_i - miss_i)^2 \tag{3.5}$$

En este proyecto se ha optado por aplicar 3 tipos de selectores diferentes: CFS, Wrapper y Relief. El motivo es que son técnicas que buscan combinaciones diferentes de los atributos, teniendo en cuenta la posible influencia de parámetros que parecen no significativo de forma individual, pero junto a otros mejoran la separabilidad de las muestras.

Capítulo 4

Procedimiento y metodología

La meta principal del presente trabajo es el desarrollo de una herramienta que permita determinar si un miARN puede regular la actividad de un ARNm. Esta tarea la llevará a cabo un clasificador que debe ser capaz de diferenciar entre dianas válidas y no válidas. Esta capacidad de discernimiento se desarrollará en base a un conjunto de datos que se tomará como referencia, lo que se llama *conjunto de entrenamiento*, y será verificado por otros datos no contemplados en él (*conjunto de validación*). Dado que las IMAs están presente en multitud de organismos, en este proyecto nos hemos centrando en la especie *homo sapiens*. A continuación presentaremos las herramientas empleadas para elaborar los datos de entrenamiento y verificación del clasificador, así como especificaremos el proceso seguido para obtenerlos.

4.1. Herramientas informáticas

La generación de los datos necesarios para desarrollar un clasificador que distinga, con un alto grado de fiabilidad, las subsecuencias dianas de un miARN, se ha realizado empleando 5 herramientas diferentes bajo la distribución GNU/Linux llamada Ubuntu 16.04: Scopus [1], Java, MySQL, ViennaRNA Package [Lorenz et al., 2011] y Weka [Wass, 2007].

4.1.1. Scopus

La ingente cantidad de información disponible en Internet, así como la falta de garantía de la calidad de la misma, dificulta notablemente la búsqueda de información sobre las interacciones entre miARN y ARNm. Por este motivo existen diversas bases de datos con información sobre artículos científicos que proporciona un alto grado de fiabilidad, al hacer referencia principalmente a revistas y congresos científicos indexados. *Scopus* es una de estas herramientas y la elegida para buscar información en este trabajo (figura 4.1). Ésta está desarrollada como Web y contiene alrededor de 22000 artículos de diversas áreas temáticas (biología, tecnología, medicina, ciencias sociales, etc.) procedente de más de 5000 editoriales internacionales, permitiendo búsqueda por primer autor, por otros autores, palabras claves, resumen, titulo, afiliación, ISSN, DOI, conferencia, etc. Asimismo, *Scopus* también ofrece información sobre autores como afiliaciones, otras publicaciones del mismo autor, referencias y número

[1] https://www.scopus.com

de citas del documento, etcétera e implementa, para usuarios registrados, un sistema de alerta sobre cambios en perfiles concretos. De este modo, esta herramienta ha sido la que ha proporcionado las referencias empleadas en el presente proyecto.

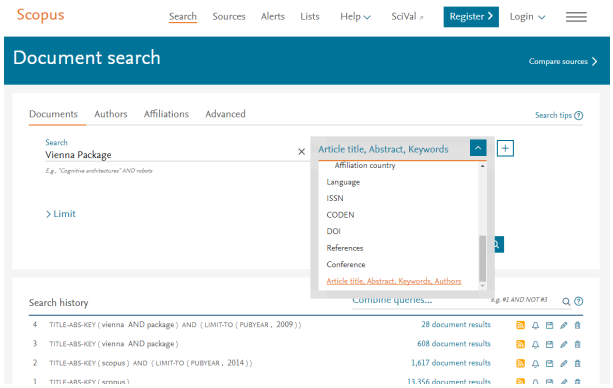


Figura 4.1: Interfaz Web de Scopus.

4.1.2. Java

La realización de tareas por máquinas como las computadoras debe ser especificada como una secuencia de código binario que le indica qué acciones y en qué orden deben realizarse. Esta codificación es difícilmente interpretable por el ser humano, por lo que se desarrollaron lenguajes formales que permiten abstraerse al programador del código binario y generar un texto más próximo a la compresión humana. Existen multitud de lenguajes de programación, tales como, C, C++, C#, Python, Ruby, Pascal, etc. De entre todos estos se ha optado por Java, un lenguaje de programación de propósito general, concurrente, orientado a objetos, cuyo diseño minimiza las dependencias con la máquina sobre la que se ejecuta gracias a lo que se llama Máquina Virtual Java (JVM). El código generado en Java es codificado en archivos intermedios en formato llamado bytecode que es tomado por la JVM para ejecutar las tareas especificadas en él, traduciendo dicha información en código binario especifico de la máquina sobre la que se está ejecutando. De este modo, Java mediante el empleo de la JVM, consigue que el programa sea escrito una única vez y se ejecute en cualquier dispositivo sin necesidad de recompilarlo en un nuevo sistema. Esta capacidad hizo que esta fuera la opción seleccionada para el proyecto. Concretamente, el trabajo ha sido desarrollado sobre la versión 1.7.

4.1.3. MySQL

MySQL es un gestor de base de datos (DB) relacionales basado en SQL, multiplataforma (FreeBSD, GNU/Linux, Windows, Mac OS X, etc.) y de código abierto bajo licencia dual GPL/comercial, inicialmente desarrollado por Sun Microsystems, y continuado después por Oracle Corporation. La consulta de información en este gestor es muy rápida, por lo que la hace ideal para webs, pero pueden darse problemas de integridad en entornos de alta concurrencia en la actualización de información. De este modo, su uso está muy extendido en aplicaciones web como Wikipedia, Wordpress, Drupal, etc.

4.1.4. ViennaRNA Package

La interacción entre pares de bases de nucleótidos se conoce como estructura secundaria (ES) que puede calcularse para un polímero o entre 2 secuencias. El paquete ViennaRNA [Lorenz et al., 2011] es un conjunto de aplicaciones y librerías escrita en C para la predicción y comparación de la ES de 2 secuencias de nucleótidos. Ésta proporciona diversos parámetros: el alineamiento entre los polímeros, la mínima energía libre de plegamiento (MFE) [Zuker, 1989], información sobre el ensamblaje termodinámico, etc. Del conjunto de herramientas que contiene el paquete ViennaRNA, para este proyecto se ha empleado el software *RNAcofold* que proporciona la información de la ES de secuencias de dímeros de ARN o ADN de una manera sencilla, basta con pasarle las 2 cadenas de nucleótidos unidas por el símbolo & para que se calcule la estructura, sin necesidad de usar secuencias ficticias de conectores. En la figura 4.2 podemos ver un ejemplo en el que se muestra el alineamiento de las secuencias y la energía mínima libre.

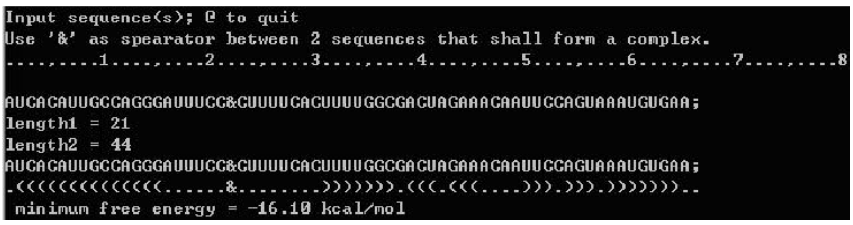


Figura 4.2: Ejemplo de uso de RNAcofold. En el alineamiento, los puntos indican nucleótidos que están desemparejados, y la apertura y cierre de paréntesis hacen referencia a la unión de 2 nucleótidos.

4.1.5. Weka

Weka es un completo software y librería libre desarrollado en Java bajo la licencia GNU que proporciona una gran cantidad de técnicas ML y FS diferentes [Wass, 2007]. Esta herramienta permite al usuario llevar a cabo análisis y representación gráfica de los datos, crear modelos predictivos y añadir nuevas extensiones. Weka proporciona 5 secciones diferentes en las que operar con datos para su clasificación y análisis: *Explorer*, *Experimenter*, *KnowleadgeFlow*, *Workbench* y *Simple CLI* (figura 4.3). Centrándonos en la sección *Explorer* (que usaremos en análisis posteriores), el usuario puede importar información desde diferentes fuentes (ficheros en diversos formatos, URL, DB o generar valores artificiales), aplicarle una amplia gama de técnicas de preprocesamiento de datos y observar un resumen gráfico y numérico de los mismos. Con los datos incorporados al sistema, es posible aplicarles técnicas de clasificación, algoritmos de selección de características y obtener un resumen detallado de los mismos, como por ejemplo, matriz de confusión, precisión de la clasificación, sensibilidad, especificidad, Kappa, etc.

4.2. Materiales

La generación de diversos clasificadores requiere de un repositorio con la información que se va a emplear en el desarrollo de los mismos. Para el correcto funcionamiento de las técnicas ML se necesita tanto casos validados de interacción entre

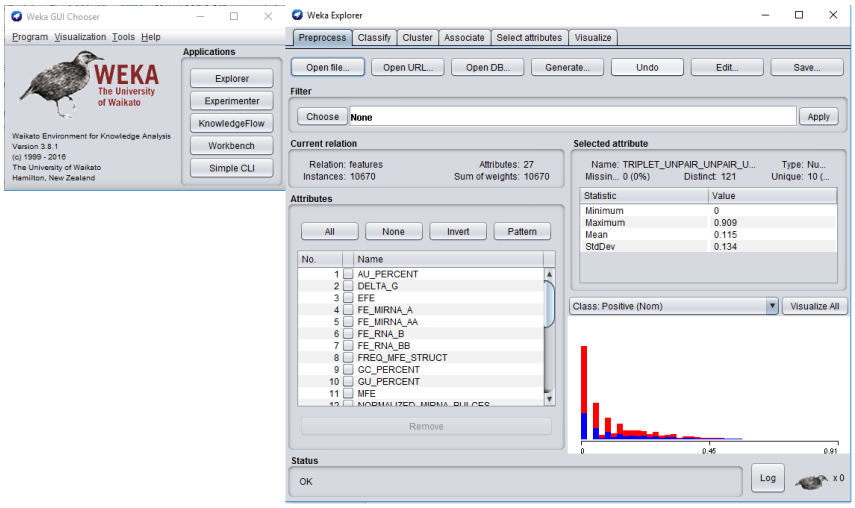


Figura 4.3: Interfaz gráfica de Weka.

miARN y ARNm, es decir, casos positivos donde la acción de un gen es controlada por un miARN, como casos confirmados de no regulación (casos negativos). De esta forma, es necesario obtener las secuencias de los miARN, las subsecuencias del gen donde se une y determinar si se trata de una interacción real.

TarBase[2] [Vlachos et al., 2015] nos proporciona un conjunto de información en formato CSV sobre interacciones validadas por expertos sobre diferentes organismos (homo sapiens, ratón, pez cebra, etc.). Se trata de un fichero que contiene información sobre más de 600000 casos donde se identifica la interacción que se valida (el miARN, el gen, especie, tipo de experimento, línea celular, etc.) y si se trata de un caso positivo o negativo de regulación, de forma que podemos verificar que los datos que se generarán son correctos.

Los información extraída de TarBase únicamente nos indica si se trata de regulación positiva o negativa, pero no nos proporciona la secuencia ni del miARN ni la subsecuencia de nucleótidos del ARNm donde se une. Las diferentes secuencias de miARN se pueden obtener de *mirBase*. Este repositorio nos facilita información diversa, entre la que destaca las secuencia completas de los miARN maduros de diferentes especies en formato FASTA[3]. Por otro lado, las secuencias completas de genes asociadas a la especie *homo sapiens* puede obtenerse de Ensembl[4] en formato FASTA, donde se identifica el gen, se enumera su transcriptor y la secuencia de nucleótidos que lo componen. Este archivo, junto con TarBase, se empleará para genera los casos de interacciones negativas. Los casos positivos siguen siendo un reto, ya que en una misma secuencia de un gen puede haber más de una posición donde se puede unir el miARN, por lo que se precisa de otra fuente que nos indique exactamente en qué segmento del ARNm se une.

Existen diversos estudios que proporcionan información sobre interacciones positivas y validadas de IMAs donde ponen a disposición de los lectores ambas secuencias (miARN y ARNm) [Lu and Leslie, 2016, Grosswendt et al., 2014, Helwak et al., 2013]. En este trabajo, los datos de regulación positiva se han obtenido del material suple-

[2] *carolina.imis.athena-innovation.gr/diana_tools/downloads/5cf98644b7b3956d3acf77f47c875f36/tarbase_data.tar.gz*

[3] ftp://mirbase.org/pub/mirbase/CURRENT/mature.fa.gz

[4] ftp://ftp.ensembl.org/pub/release-87/fasta/homo_sapiens/cdna/Homo_sapiens.GRCh38.cdna.all.fa.gz

mentario ("Data S1") del artículo [Helwak et al., 2013] debido a la claridad con la que se presentaba la información. Este archivo contiene entorno a 18000 muestras donde se identifica el miARN, su secuencia, el ARNm que regula y su subsecuencia de nucleótidos extendida 25 unidades.

4.3. Generación de la base de datos

El proceso de generación de la información necesaria para entrenar a los clasificadores se realiza en 8 etapas:

1. Creación de las tablas en la DB.
2. Incorporación de las secuencias completas de los genes de la especie *homo sapiens* obtenidos de Ensembl.
3. Inserción de las secuencias de los miARN. Las cadenas fueron insertadas de forma que la región semilla quedara ubicada al final (como en la figura 1.2).
4. Registro de los casos positivos obtenidos de [Helwak et al., 2013].
5. Generación e inclusión de ejemplos negativos de interacciones empleando para ello la información de TarBase y las secuencias completas de los ARNm.
6. Eliminación de los miARN y ARNm a los que no se les ha asociado ninguna interacción.
7. Eliminación de las secuencias completas de los genes una vez se han generado los casos negativos. Asimismo, se eliminan las subsecuencias de nucleótidos repetidas y las interacciones que tuvieran asociadas.
8. Reenumeración de las interacciones de forma correcta y correlativa, de forma que los números mayores a 0 se asocian a los casos positivos, y lo menores a los casos negativos.

De esta secuencia de pasos, detallaremos los puntos 1,4 y 5. Los restantes se pueden analizar en el código Java que se adjunta en el proyecto.

4.3.1. Creación de las tablas

Tras la revisión de la bibliografía empleada en el TFM, se determinó que 3 tablas en MySQL serían suficientes para contener toda la información necesaria: datos de los ARNm, secuencias de miARN e interacciones.

Tabla de secuencias de ARNm

Tiene como objetivo contener toda la información relevante de un gen, de forma que se identifique de inequívocamente y tengamos disponible su secuencia de nucleótidos completa o una subsecuencia de la misma. La instrucción SQL empleada para crear esta tabla es:

CREATE TABLE RNA(Id VARCHAR(40), Transcription_Id VARCHAR(20), Name VARCHAR(50), Version INT NOT NULL, Sequence LONGTEXT, CompletedSequence BOOL, PRIMARY KEY(Id, Transcription_Id, Version))

donde el significado de los campos es el siguiente:

- *Id*: identificador único del gen.
- *Transcription_Id*: identificador de la versión de transcripción del gen.
- *Name*: nombre del gen.
- *Version*: versión del gen. Este campo tiene como objetivo diferenciar diferentes subsecuencias o segmentos del mismo gen sobre los que interactúan los miARN. Asimismo, también permite incorporar las diferentes variantes de una secuencia que viene codificada usando los símbolos de la *Unión Internacional de Química Pura y Aplicada* (UIQPA) [5]. Los valores de versiones negativos se han empleado únicamente para secuencias completas del gen, mientras que los positivos se dejan para segmentos del gen.
- *Sequence*: secuencia de nucleótidos.
- *CompletedSequence*: indica si la secuencia de nucleótidos del registro es un segmento del gen (*falso*) o se trata del gen completo (*verdadero*).

Tabla de secuencias de miARN

El objetivo de esta tabla es registra los datos relevantes asociados a un miARN. La instrucción SQL empleada para crear esta tabla es:

CREATE TABLE miRNA(Id VARCHAR(20), Accession_No VARCHAR(20), Sequence VARCHAR(50), PRIMARY KEY(Id, Accession_No))

donde los campos indican:

- *Id*: nombre del miARN.
- *Accession_No*: identificador secundario. El nombre del miARN puede cambiar en un futuro, sin embargo este número es estable y no varía.
- *Sequence*: secuencia de nucleótidos.

Tabla de interacciones

Contiene todo la información relevante a la hora de relacionar un miARN con la sección del gen al que regula. La instrucción SQL empleada para crear esta tabla es:

CREATE TABLE miRNA_RNA_interaction(miRNA_Id VARCHAR(20), RNA_Id VARCHAR(40), Transcription_Id VARCHAR(20), Version INT NOT NULL, No INT NOT NULL, PositiveCase BOOL, UpDown INT NOT NULL, PRIMARY KEY(miRNA_Id,

[5] $https://www.bioinformatics.org/sms/iupac.html$

Transcription_Id, Version, No, PositiveCase), FOREIGN KEY (miRNA_Id) REFERENCES miRNA(Id), FOREIGN KEY (RNA_Id, Transcription_Id, Version) REFERENCES RNA(Id, Transcription_Id, Version)

donde los campos se identifican como:

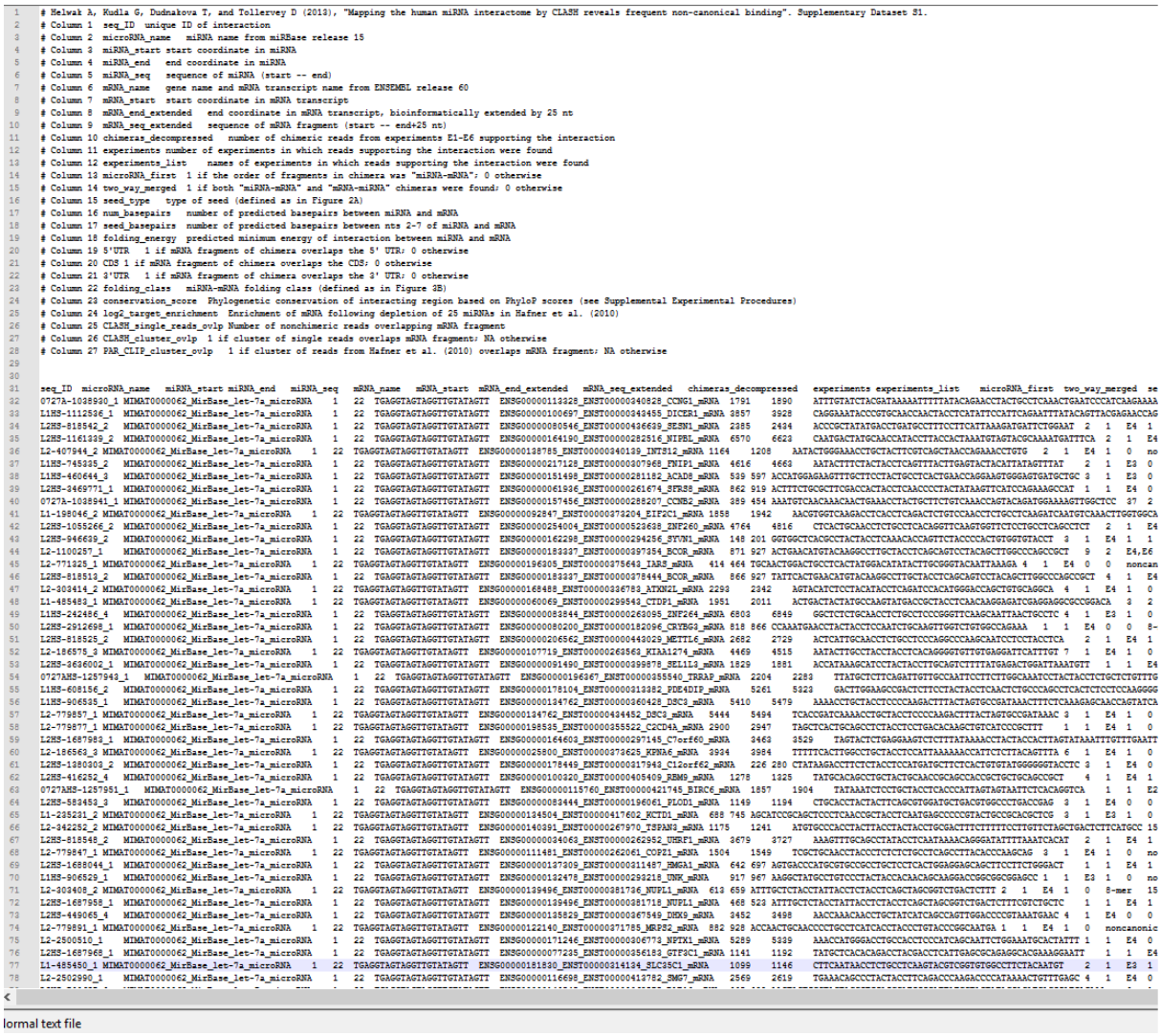
- *miRNA_Id*: *Id* del miARN que interviene en la interacción.
- *RNA_Id*: *Id* del gen con el que interactúa.
- *Transcription_Id*: identificador del transcriptor del gen.
- *Version*: versión del gen de la interacción, es decir, identifica el segmento del ARNm al que se une el miARN.
- *No*: numeración de la interacción. Este campo se empleará en el proceso de creación de los conjuntos de entrenamiento y verificación. Los valores positivos se han empleado para los casos de regulación positiva, mientras que los valores menores de 0 se han dejado para los casos negativos.
- *PositiveCase*: indica si se trata de una interacción real, es decir, que el miARN regula la actividad del gen o no.
- *UpDown*: indica el tipo de regulación de la interacciones positivas. Esta información está contenida en TarBase.

Especificado el conjunto de tablas que se van a emplear en el proyecto, pasaremos a continuación a insertar información en éstas.

4.3.2. Inserción de casos positivos

En este trabajo, los datos de regulación positiva se han obtenido del material suplementario ("Data S1") del artículo [Helwak et al., 2013]. Éstos se validaron con la información TarBase, de forma que se descarta las interacciones clasificadas como negativas y aquellas donde la regulación (columna UpDown del fichero de TarBase) sean contradictorias. En la figura 4.4 puede verse algunas de las líneas del fichero. Si se observa con detenimiento, se puede ver que contiene toda la información relevante para rellenar las 3 tablas de la DB, dividiéndose en 3 partes para cada línea del archivo:

1. Se inserta en la tabla del *ARNm* la información del ARN extraída del fichero. Se obtuvieron el Id del gen, su transcriptor, nombre y subsecuencia de nucleótidos. En este proceso se comprueba si la secuencia ya está en la DB, en caso positivo se aborta el proceso y se obtiene el número de segmento (versión) de la secuencia ya existen, de lo contrario, se inserta asignándole el siguiente número de segmento (versión).
2. Tras insertar la secuencia en la tabla *ARNm*, procedemos con el miARN. Del archivo obtendremos su secuencia, y los identificadores *Id* y *Accession_No*. Ésta información se insertará en caso de que no esté ya registrada.



```
# Helwak A, Kudla G, Dudnakova T, and Tollervey D (2013), "Mapping the human miRNA interactome by CLASH reveals frequent non-canonical binding". Supplementary Dataset S1.
# Column 1  seq_ID  unique ID of interaction
# Column 2  microRNA_name   miRNA name from miRBase release 15
# Column 3  miRNA_start start coordinate in miRNA
# Column 4  miRNA_end   end coordinate in miRNA
# Column 5  miRNA_seq   sequence of miRNA (start -- end)
# Column 6  mRNA_name   gene name and mRNA transcript name from ENSEMBL release 60
# Column 7  mRNA_start  start coordinate in mRNA transcript
# Column 8  mRNA_end_extended   end coordinate in mRNA transcript, bioinformatically extended by 25 nt
# Column 9  mRNA_seq_extended   sequence of mRNA fragment (start -- end+25 nt)
# Column 10 chimeras_decompressed   number of chimeric reads from experiments E1-E6 supporting the interaction
# Column 11 experiments number of experiments in which reads supporting the interaction were found
# Column 12 experiments_list    names of experiments in which reads supporting the interaction were found
# Column 13 microRNA_first  1 if the order of fragments in chimera was "miRNA-mRNA"; 0 otherwise
# Column 14 two_way_merged  1 if both "miRNA-mRNA" and "mRNA-miRNA" chimeras were found; 0 otherwise
# Column 15 seed_type   type of seed (defined as in Figure 2A)
# Column 16 num_basepairs   number of predicted basepairs between miRNA and mRNA
# Column 17 seed_basepairs  number of predicted basepairs between nts 2-7 of miRNA and mRNA
# Column 18 folding_energy  predicted minimum energy of interaction between miRNA and mRNA
# Column 19 5'UTR   1 if mRNA fragment of chimera overlaps the 5' UTR; 0 otherwise
# Column 20 CDS 1 if mRNA fragment of chimera overlaps the CDS; 0 otherwise
# Column 21 3'UTR   1 if mRNA fragment of chimera overlaps the 3' UTR; 0 otherwise
# Column 22 folding_class   miRNA-mRNA folding class (defined as in Figure 3B)
# Column 23 conservation_score  Phylogenetic conservation of interacting region based on PhyloP scores (see Supplemental Experimental Procedures)
# Column 24 log2_target_enrichment  Enrichment of mRNA following depletion of 25 miRNAs in Hafner et al. (2010)
# Column 25 CLASH_single_reads_ovlp Number of nonchimeric reads overlapping mRNA fragment
# Column 26 CLASH_cluster_ovlp  1 if cluster of single reads overlaps mRNA fragment; NA otherwise
# Column 27 PAR_CLIP_cluster_ovlp   1 if cluster of reads from Hafner et al. (2010) overlaps mRNA fragment; NA otherwise


seq_ID  microRNA_name   miRNA_start miRNA_end   miRNA_seq   mRNA_name   mRNA_start  mRNA_end_extended   mRNA_seq_extended   chimeras_decompressed   experiments experiments_list    microRNA_first  two_way_merged  se
0727A-1038930_1 MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000113328_ENST00000340828_CCNG1_mRNA  1791    1890    ATTTGTATCTACGATAAAAATTTTTATACAGAACCTACTGCCTCAAACTGAATCCCATCAAGAAAA
L1HS-1112536_1  MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000100697_ENST00000343455_DICER1_mRNA 3857    3928    CAGGAAATACCCGTGCAACCAACTACCTCATATTCCATTCAGAATTTATACAGTTACGAGAACCAG
L2HS-818542_2   MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000080546_ENST00000436639_SEPN1_mRNA  2385    2424    ACCCGCTATATGACCTGATGCCTTTCCTTCATTAAAGATGATTCTGGAAT  2   1   E4  1
L2HS-1161399_2  MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000164190_ENST00000282516_NIPBL_mRNA  6570    6623    CAATGACTATGCAACCATACCTTACCACTAAATGTAGTACGCAAAATGATTTCA  2   1   E4
L2-407944_2 MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000138785_ENST00000340139_INTS12_mRNA 1164    1208    AATACTGGGAAACCTGCTACTTCGTCAGCTAACCAGAAACCTGTG   2   1   E4  1   0   no
L1HS-745325_2   MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000217128_ENST00000307968_FNIP1_mRNA  4616    4663    AATACTTTCTACTACCTCAGTTTACTTGAGTACTACATTATAGTTTAT    2   1   E3  0
L1HS-460644_3   MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000151498_ENST00000281182_ACAD8_mRNA  539 597 ACCATGGAGAAGTTTGCTTCCTACTGCCTCACTGAACCAGGAAGTGGGAGTGATGCTGC   3   1   E3  0
L2HS-3469771_1  MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000061936_ENST00000261674_SFRS8_mRNA  862 919 ACTTTCTGCGCTTCGACCACTACCTCAACCCCTACTATAAGTTCATCCAGAAAGCCAT  1   1   E4  0
0727A-1038941_1 MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000157456_ENST00000288207_CCNB2_mRNA  389 454 AAATGTCAACAAACAACTGAAACCTACTGCTTCTGTCAAACCAGTACAGATGGAAAAGTTGGCTCC  37  2
L1-198046_2 MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000092847_ENST00000372204_EIF2C1_mRNA 1858    1942    AACGTGGTCAAGACCTCACCTCAGACTCTGTCCAACCTCTGCCTCAAGATCAATGTCAAACTTGGTGGCA
L2HS-1055266_2  MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000254004_ENST00000523638_ZNF260_mRNA 4764    4816    CTCACTGCAACCCTCTGCCTCACAGGTTCAAGTGGTTCTCCTGCCTCAGCCTCT  2   1   E4
L2HS-946629_2   MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000162298_ENST00000294256_SYVN1_mRNA  148 201 GGTGGCTCACGCCTACTACCTCAAACACCAGTTCTACCCCACTGTGGTGTACCT  3   1   E4  1   1
L2-1100257_1    MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000183337_ENST00000397354_BCOR_mRNA   871 927 ACTGAACATGTACAAGGCCTTGCTACCTCAGCAGTCCTACAGCTTGGCCCAGCCGCT  9   2   E4,E6
L2-771325_1 MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000196305_ENST00000375643_IAR3_mRNA   414 464 TGCAACTGGACTGCCTCACTATGGACATATACTTGCGGGTACAATTAAAGA 4   1   E4  0   0   noncan
L2HS-818513_2   MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000183337_ENST00000378444_BCOR_mRNA   866 927 TATTCACTGAACATGTACAAGGCCTTGCTACCTCAGCAGTCCTACAGCTTGGCCCAGCCGCT  4   1   E4
L2-303414_2 MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000168488_ENST00000336783_ATXN2L_mRNA 2293    2342    AGTACATCTCCTACATACCTCAGATCCACATGGGACCAGCTGTGCAGGCA   4   1   E4  1   0
L1-485483_1 MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000060069_ENST00000299543_CTDP1_mRNA  1951    2011    ACTGACTACTATGCCAAGTATGACCGCTACCTCAACAAGGAGATCGAGGAGGCGCCGGACA   3   2
L1HS-242486_4   MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000085344_ENST00000263095_ENPP4_mRNA  6803    6849    GGCTCTCTGCAACCCTCTGCCTCCCGGGTTCAAGCAATTAACTGCCTC    4   1   E3  1   0
L2HS-2912498_1  MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000080200_ENST00000182066_CYTH2_mRNA  818 866 CCAAATGAACCTACTACCTCCAATCTGCAAGTTGGTCTGTGGCCAGAAA  1   1   E4  0   0   8-
L2HS-818525_2   MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000206562_ENST00000443029_METTL6_mRNA 2682    2729    ACTCATTGCAACCTCTGCCTCCCAGGCCCAAGCAATCCTCCTCAACCTCA   2   1   E4  1
L2-186575_3 MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000107719_ENST00000263563_KIAA1274_mRNA   4469    4515    AATACTTGCCTACCTACCTCACAGGGGTGTTGTGAGGATTCATTTGT 7   1   E4  1   0
L2HS-3636002_1  MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000091490_ENST00000399878_SEL1L3_mRNA 1829    1881    ACCATAAAGCATCCTACTACCTTGCAGTCTTTTATGAGACTGGATTAAATGTT   1   1   E4
0727AH3-1257943_1   MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000196367_ENST00000355540_TRRAP_mRNA  2204    2283    TTATGCTCTTCAGATTGTTGCCAATTCCTTCTTGGCAAATCCTACTACCTCTGCTCTGTTTG
L1HS-608156_2   MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000178104_ENST00000313382_PDE4DIP_mRNA   5261    5323    GACTTGGAAGCCGACTCTTCCTACTACCTCAACTCTGCCCAGCCTCACTCTCCTCCAAGGGG
L1HS-906535_1   MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000134762_ENST00000360428_DSC3_mRNA   5410    5479    AAAACCTGCTACCTCCCCAAGACTTTACTAGTGCCGATAAACTTTCTCAAAGAGCAACCAGTATCA
L2-779857_1 MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000134762_ENST00000434452_DSC3_mRNA   5444    5494    TCACCGATCAAAACCTGCTACCTCCCCAAGACTTTACTAGTGCCGATAAAC 3   1   E4  1   0
L2-779877_1 MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000198535_ENST00000355522_C2CD4A_mRNA 2900    2947    TAGCTCACTGCAGCCTCTACCTCCTGACACAAGCTGTCATCCCGCTTT    1   1   E4  1   0
L2HS-1687983_1  MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000164603_ENST00000297145_C7orf60_mRNA   3463    3529    TAGTACTCTGAGGAAGTCTCTTTATAAAACCTACTACCACTTAGTATAAATTTGTTTGAATT
L2-186563_3 MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000025800_ENST00000373625_KFNA6_mRNA  3924    3984    TTTTTCACTTGGCCTGCTACCTCCATTAAAAAACCATTCTCTTACAGTTTA 6   1   E4  1   0
L2HS-1380303_2  MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000178449_ENST00000317943_C12orf62_mRNA  226 280 CTATAAGACCTTCTCTACCTCCATGATGCTTCTCACTGTGTATGGGGGGTACCTC 3   1   E4  0
L2HS-416282_4   MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000100320_ENST00000405409_REM9_mRNA   1278    1325    TATGCACAGCCTGCTACTGCAACCGCAGCCACCGCTGCTGCAGCCGCT    4   1   E4  1
0727AH3-1257951_1   MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000115760_ENST00000421745_BIRC6_mRNA  1857    1904    TATAAATCTCCTGCTACCTCACCCATTAGTAGTAATTCTCACAGGTCA    1   1   E2
L2HS-583483_3   MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000083444_ENST00000196061_PLOD1_mRNA  1149    1194    CTGCACCTACTACTTCAGCGTGGATGCTGACGTGGCCCTGACCGAG  3   1   E4  0   0
L1-235231_2 MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000134504_ENST00000417602_WCTD1_mRNA  688 745 AGCATCCGCAGCTCCCTCAACCGCTACCTCAATGAGCCCCCGTACTGCCGCACGCTCG    3   1   E3  1   0
L2-342252_2 MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000140391_ENST00000267570_TSPAN3_mRNA 1175    1241    ATGTGCCCACCTACTTACCTACTACCTGCGACTTTCTTTTCCTTGTCTACGCTGACTCTTCATGCC 15
L2HS-818548_2   MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000004063_ENST00000262352_UHRF1_mRNA  3679    3727    AAAGTTTGCAGCCTATACCTCAATAAAACAGGGATATTTTAAATCACAT   2   1   E4  1
L2-779847_1 MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000111481_ENST00000262061_COPZ1_mRNA  1504    1549    TCGCTGCAACCTACCCCTCTCTCTGCCTCAGCCTTACACCCAAGCAG 3   1   E4  1   0   no
L2HS-1688044_1  MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000137309_ENST00000311487_HMGA1_mRNA  642 697 AGTGACCCATGCGTGCCGCCTGCTCCTCACTGGAGGAGCAGCTTCCTTCTGGGACT  1   1   E4  1
L1HS-906529_1   MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000132478_ENST00000293218_UNK_mRNA    917 967 AAGGCTATGCCTGTCCCCTACTACCACAACAGCAAGGACCGGCGGCGGAGCC 1   1   E3  1   0   no
L2-303408_2 MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000139496_ENST00000381736_NUPL1_mRNA  613 659 ATTTGCTCTACCTATTACCTCTACCTCAGCTAGCGGTCTGACTCTTT 2   1   E4  1   0   8-mer   15
L2HS-1687958_1  MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000139496_ENST00000381718_NUPL1_mRNA  468 523 ATTTGCTCTACCTATTACCTCTACCTCAGCTAGCGGTCTGACTCTTTCGTCTGCTC   1   1   E4  1
L2HS-449065_4   MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000135829_ENST00000367549_DHX9_mRNA   3452    3498    AACCAAACAACCTGCTATCATCAGCCAGTTGGACCCCGTAAATGAAC 4   1   E4  0   0
L2-779851_1 MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000122140_ENST00000371785_MRPS2_mRNA  882 928 ACCAACTGCAACCCCTGCCTCATCACCTACCCTGTACCCGGCAATGA 1   1   E4  1   0   noncanonic
L2-2500510_1    MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000171246_ENST00000306773_NFTX1_mRNA  5289    5339    AAACCATGGGACCTGCCACCTCCCATCAGCAATTCTGGAAATGCACTATTT 1   1   E4  0
L2HS-1687968_1  MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000077235_ENST00000356183_GTF3C1_mRNA 1141    1192    TATGCTCACACAGACCTACAGACCTACTGAGCTCAGAGGCACGAAAGGAATT    1   1   E4
L1-485450_1 MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000181830_ENST00000314134_SLC35C1_mRNA   1099    1146    CTTCAATAACCTCTGCCTCAAGTACGTCGGTGTGGCCTTCTACAATGT    2   1   E3  1
L2-2502990_1    MIMAT0000062_MirBase_let-7a_microRNA    1   22  TGAGGTAGTAGGTTGTATAGTT  ENSG00000116698_ENST00000413782_SMG7_mRNA   2569    2619    TGAAACAGCCCTACTACCTTCAGACCCAAGACCCCATAAAACTGTTTGAGC 4   1   E4  0
<
lormal text file
```

Figura 4.4: Líneas del fichero de casos positivos.

3. El último punto del proceso registra la interacción, teniendo en cuenta las diferentes versiones del ARNm que estén presentes en la DB (descodificación de los símbolos de UIQPA), indicando de que se trata de un caso positivo y asignándole el número de interacción siguiente disponible y mayor que 0.

4.3.3. Generación de casos negativos

La localización de muestras de secuencias de falsas interacciones entre miARN y ARNm no es sencilla. De hecho, no ha sido posible detectar ningún repositorio que proporcione esta información. Por ello, se ha usado algunas de las interacciones negativas contenidas en TarBase, concretamente las relacionadas con *homo sapiens* y cuyo identificador siguen el formato de Ensembl. Con esta información, la tabla de miARN y las secuencias completas de ARNm contenidas en la tabla de ARNm se genera un conjunto inicial de interacciones negativas donde se relaciona las secuencias completas de los genes y los miARN. Con esta información, se procede a dividir las secuencias en posibles segmentos candidatos aplicando como estrategia de búsqueda una ventana deslizante (figura 4.5). El proceso es como sigue:

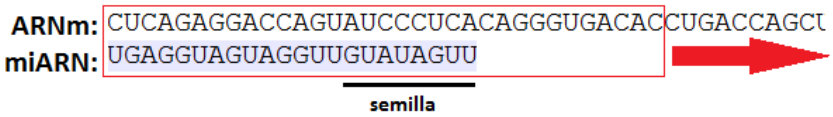


Figura 4.5: Generación de casos negativos. Ejemplo de ventana deslizante.

1. Inicialmente, se toma un segmento de ARNm igual a la longitud del miARN más 15 nucleótidos adicionales. Estos polímeros extras se añaden en base a 2 criterios: 1) a que se comprobó experimentalmente que en los casos de interacciones positivas, el alineamiento generado por ViennaRNA Package era mejor cuando la cadena del ARNm es un poco mayor que el miARN, y 2) que en [Helwak et al., 2013] se proporciona la subsecuencia del sitio de unión extendida en 25 nucleótidos de más.

2. Con este segmento generamos la entrada al programa *RNAcofold* de forma que en primer lugar va el miARN y luego la subsecuencia del ARNm pero en orden inverso. De esta forma, las entrada para las secuencias de la figura 4.5 sería: *RNAcofold -a "UGAGGUAGUAGGUUGUAUAGUU&GUCCACAGUGGGACACUCCCUAUGACCAGGAGACUC"* (la región semilla del miARN está al final de la cadena, leyendo de izquierda a derecha).

3. De la salida del punto anterior se determina si el segmento es un candidato a interacción negativa. Varios son los parámetros que nos proporciona el programa, de los cuales usaremos 2 de ellos: el alineamiento para determinar el tipo de coincidencia de semilla y la mínima energía libre de plegamiento (MFE). Dos situaciones son posibles:

   - Se descarta el segmento y se actualiza la subsecuencia desplazando la ventana un nucleótido, es decir, el primero se elimina y se añade al final el siguiente nucleótido del ARNm. El *criterio de rechazo* debe cumplir al menos una de las 2 condiciones siguientes: el valor de MFE es superior a -20 o el tipo de unión de la región semilla no es ninguno de los que se especifico en la sección 1.1.1. Con la nueva secuencia se vuelve a ejecutar *RNAcofold*.
   - La subsecuencia se clasifica como candidato a interacción negativa ($MFE < -20$ y la unión de la semilla es de uno de los tipos de la sección 1.1.1). En esta circunstancia pueden darse de nuevo 2 situaciones:
     - Al desplazar la ventana de segmentos del ARNm se cumple el *criterio de rechazo*. En esta situación insertamos la interacción en la base de datos, incorporando la subsecuencia en la tabla de ARNm con su correcto valor de versión (valor mayor a 0, ya que los valores negativos se dejaron para secuencias completas del gen).
     - Al actualizar el segmento del ARNm, desplazando la ventana 1 nucleótido, se siguen sin cumplir el *criterio de rechazo*. En esta situación si el tipo de unión de semilla contiene más uniones entre nucleótidos, actualizamos el candidato con este nuevo segmento, de lo contrario mantenemos el anterior (en caso de semillas no canónicas este criterio no se aplica, sino que se mantiene el candidato). En cualquiera de los casos,

seguimos desplazando la ventana hasta que se cumpla el criterio de rechazo, tras lo cual rellenamos la información de la interacción negativa en la DB como se explicó en el punto anterior.

Este proceso se repetirá hasta que se haya analizado la secuencia completa del ARNm. Computacionalmente, este proceso es muy costo, produciendo una gran sobrecarga de trabajo a los procesadores del computador, y a su vez, siendo un proceso muy lento, llegando a tardar alrededor de 1 semana en procesar todos los candidatos. Por ello, para acelerarlo se dividió el procesamiento en 7 hilos en Java (el computador sobre el que se desarrollo el proyecto es de 4 núcleos de 8 hilos) donde cada uno se ocupaba de procesar diferentes genes. Esto redujo el tiempo considerablemente, hasta aproximadamente 80 horas.

Durante este proceso se dieron varios problemas en la ejecución del sistema (problemas en el código, error en la conexión con al DB, apagado del computador, etc.) que supuso un notablemente retraso en la planificación del proyecto, por lo que no se puedo incorporar nuevos atributos a los clasificadores ni estudiar posibles mejoras del mismo.

4.4. Extracción de características de las interacciones miARN:ARNm

Definidas las interacciones entre las secuencias de microARN y ARN mensajeros, podemos extraer un conjunto amplio de parámetros que se emplearán para entrenar los clasificadores. Estas características son:

1. **Encaje de semilla**: como se explicó en capítulo 1, la semilla del miARN puede configurar hasta 9 tipos de encajes diferentes [Peterson et al., 2014, Yue et al., 2009, Kim et al., 2016]:

   a) Tipo 1: combinaciones perfectas de 8 nucleótidos desde la posición 2 a la 8.

   b) Tipo 2: combinaciones perfectas de 8 nucleótidos desde la posición 2 a la 8 con un nucleótido tipo A en la posición 1.

   c) Tipo 3: combinaciones perfectas de los nucleótidos desde la 2 al 7.

   d) Tipo 4: combinaciones perfectas de los nucleótidos desde la 2 al 7 y un nucleótido tipo A en la posición 1.

   e) Tipo 5: combinaciones perfectas de los nucleótidos desde la 2 al 7 o a lo sumo 1 combinación G-U.

   f) Tipo 6: combinaciones perfectas de 6 nucleótidos.

   g) Tipo 7: combinaciones perfectas de 6 nucleótidos o a lo sumo 1 combinación G-U.

   h) Tipo 8: encaje no canónico complementaria.

   i) Tipo 9: encaje no canónico centrada.

2. Composición de secuencia [Lopes et al., 2015, Yue et al., 2009]:

a) Porcentaje de pares de nucleótidos en la interacción: %A+U, %G+C y %G+U.

b) Porcentaje de protuberancias en cada secuencia de la interacción (miARN y ARNm).

c) Porcentaje de protuberancias en la región semilla tanto en el miARN como en el ARNm.

d) Porcentaje de emparejamientos de subsecuencias de 3 nucleótidos entre miARN y ARNm. Se evalúan con una ventana deslizando comprobando si entre los nucleótidos existe o no un correcto emparejamiento. En total, hay 8 casos diferentes:

- Los 3 nucleótidos están perfectamente emparejados en la interacción.
- Ninguno de ellos están unidos en la interacción.
- Sólo 1 de los 3 combina.
- Únicamente 1 de los nucleótidos no está emparejado.

e) Porcentaje de incompatibilidades de pares de bases, por ejemplo %A-C.

f) Porcentaje de incompatibilidades de pares de bases en la región semilla.

3. Características estructurales [Lopes et al., 2015, Yue et al., 2009, Kim et al., 2006]: el paquete “Vienna RNA” y la aplicación *RNAfold* calculan algunos de estos parámetros.

a) Mínima energía libre de plegamiento (MFE).

b) Energía libre de banda (EFE).

c) Energía de Gibbs (EG). Indica el equilibrio de una reacción química.

d) Energía de libre del autoplegamiento (FEA). Se calcula tanto en el miARN como en el mARN.

e) Energía de libre del plegamiento de la secuencia con una idéntica (FEAG). Se calcula tanto para el miARN como para el mARN.

Esto hace un total de 26 características a evaluar con las técnicas FS y a usar en los algoritmos de ML. No obstante, en el anexo 1 (capítulo A) se lista un conjunto de parámetros que se barajaron para en el proyecto pero que fueron descartados por falta de tiempo.

Capítulo 5

Resultados

Al finalizar el proceso de generación de información relacionada con las interacciones entre miARN y ARNm, la DB contiene 18091 casos positivos frente a 16711 de casos negativos. De estas interacciones, se han extraído 26 características (sección 4.4), ignorando aquellas interacciones que no cumplían los criterios de rechazo que se detalló en la sección 4.3.3. Tras este proceso, el resultado final son 2 ficheros en formato CSV: uno para FS y entrenamiento de los clasificadores con 10670 muestras (7021 casos positivos, 3649 casos negativos), y otro para evaluar el rendimiento de los clasificadores con 3576 (2339 casos positivos, 1237 casos negativos).

5.1. Selección de características

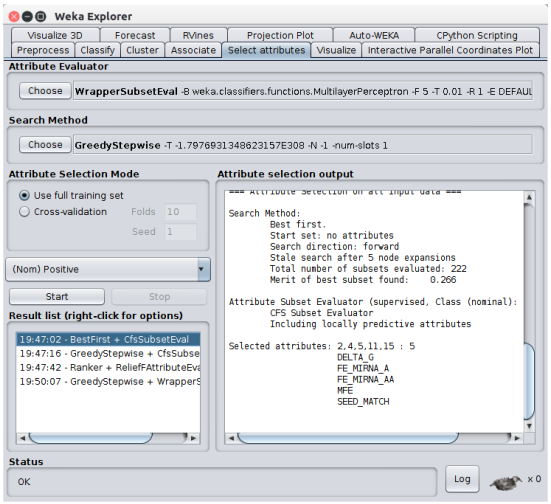
El conjunto de características obtenido han sido analizadas mediante 3 técnicas y 4 estrategias de búsqueda diferente para seleccionar los atributos más significativos (capítulo 3). Los resultados se pueden ver en la figura 5.1 y en la tabla 5.1. Estos fueron:

- *CfsSubSetEva + BestFirst*: se aplica *selección subconjuntos* (CFS) con estrategia de búsqueda *mejor el primero* (figura 5.1a). Del total de las características, se seleccionaron 5: EG, FEA del miARN, FEAG del microARN, MFE y el tipo de ensamblaje de la semilla.

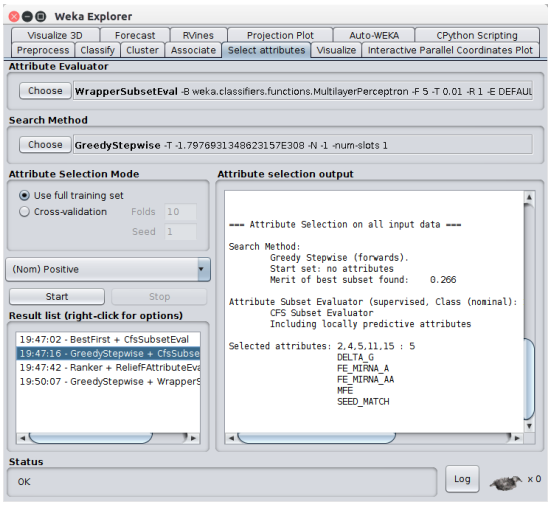
- *CfsSubSetEva + GreedyStepwise*: se emplea *selección de subconjunto* (CFS) aplicando estrategia de búsqueda *Greedy Step Wise*, en la cual se optó por búsqueda *hacia adelante* (figura 5.1b). Las características seleccionadas fueron las mismas que en el punto anterior.

- *RELIEF*: se usa la técnica *RELIEF* con estrategia de búsqueda por *ranking* (figura 5.1c). No descartó ninguna de las 26 atributos extraídas de los datos.

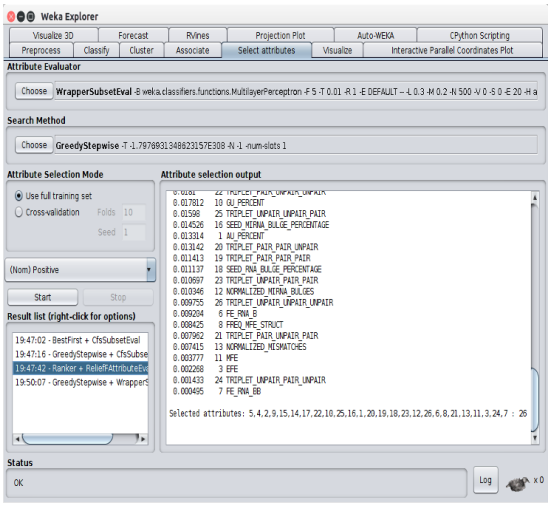
- Wrapper: algoritmo *Wrapper* empleado *SVM* para seleccionar los atributos de los datos obligando a una estratégica de búsqueda *Greedy Step Wise*, en la cual se optó por la estrategia *hacia adelante* (figura 5.1d). Seleccionó 3: EG, FEA del miARN y FEAG del miARN.



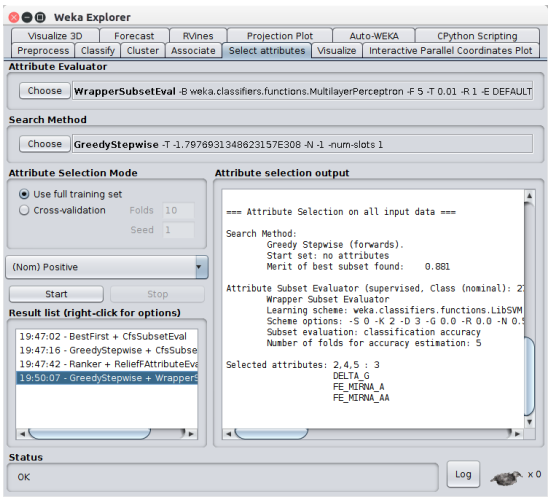
(a) Técnica: Selección de subconjunto con estrategia de búsqueda mejor el primero.



(b) Técnica: Selección de subconjunto con estrategia de búsqueda Greedy Step Wise.



(c) Técnica: RELIEF (obliga a búsqueda por ranking).



(d) Técnica: Wrapper empleando SVM para evaluar los atributos.

Figura 5.1: Resultados de los algoritmos de selección de características.

5.2. Clasificadores

Se evaluaron 2 tipos de clasificador: SVM con kernel lineal, y RN con una única capa oculta de 7 neuronas. Para cada uno de los 3 conjuntos de atributos seleccionados en la sección anterior se obtuvieron ambos clasificadores, empleando un 10-folding con estrategia de validación. Los resultados obtenidos sobre los datos de entrenamientos se muestran en las figuras 5.2 y 5.3 fueron:

- La evaluación del clasificador *ZeroR*, es decir, la estrategia de suponer que todos los casos son de interacciones positiva, proporciona una precisión del 65.80 % (figura 5.2g).
- Usando sólo los 5 parámetros de la selección CFS (EG, FEA del miARN, FEAG del miARN, MFE y el tipo de semilla), el SVM proporciona una precisión del 81.08 % (figura 5.2d), mientras que la RN da un 86.34 % (figura 5.2a).

Tabla 5.1: Parámetros seleccionados por las técnicas FS. Las filas son los atributos, mientras que las columnas identifican a la técnicas FS empleadas. El signo *X* indica que ese atributo ha sido seleccionado.

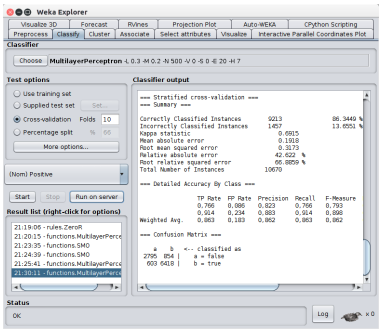
	CfsSubSetEval + BestFirst	CfsSubSetEva +GreedyStep	Relief	Wrapper
%AU			X	
EG	X	X	X	X
EFE			X	
FEA del miARN	X	X	X	X
FEAG de miARN	X	X	X	X
FEA del ARN			X	
FEAG del ARN			X	
MEFF			X	
%GC			X	
MFE	X	X	X	
%GU			X	
Protuberancias miARN			X	
% incompatibilidades			X	

	CfsSubSetEval + BestFirst	CfsSubSetEva +GreedyStep	Relief	Wrapper
Protuberancias ARN			X	
Encaje semilla	X	X	X	
Protuberancias en semilla			X	
Incompatibilidades en semilla			X	
Protuberancias en semilla			X	
Secuencias de 3: emparejamientos perfecta			X	
Secuencias de 3: desemparejamientos de todos los nucleótidos			X	
Secuencias de 3: sólo el primer nucleótido combina			X	
Secuencias de 3: sólo el segundo nucleótido combina			X	
Secuencias de 3: sólo el tercer nucleótido combina			X	
Secuencias de 3: sólo el primer nucleótido no combina			X	
Secuencias de 3: sólo el segundo nucleótido no combina			X	
Secuencias de 3: sólo el tercer nucleótido no combina			X	

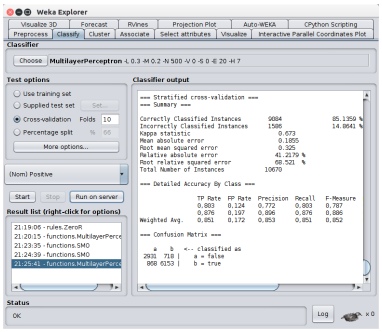
- Empleando los 26 atributos seleccionado por la técnica RELIEF, el SVM arroja un ratio de clasificación correcta del 83.45 % (figura 5.2e) frente al 85.14 % de la red neuronal (figura 5.2b).
- Por último, con sólo los 3 parámetros del método Wrapper (EG, FEA del miARN y FEAG del miARN), el SVM proporciona un 80.77 % de precisión (figura 5.2f), mientras que la red generó un 85.15 % de clasificación correcta (figura 5.2c).

En todos los casos se consigue una mejora considerable con respecto al clasificador *ZeroR* (entorno al 20 % de mejora), lo que es indicativo de que los parámetros empleados mejoran la distinción entre las clases de las interacciones. El mejor de los casos se logra con una red neuronal para los 5 atributos seleccionados con CFS en el conjunto de entrenamiento. Es por ello, que será esta red la que se use en el programa Java (sección 5.3). Para confirmar el resultado arrojado en la fase de entrenamiento, se ha evaluado el clasificador seleccionado empleando el segundo de los ficheros generados que se citó al comienzo de este capítulo y que está configurado por muestras no contendidas en el fichero de entrenamiento. La evaluación resultante se muestra en la figura 5.4. Se puede ver que se obtiene un 86.05 % de precisión en la clasificación de las muestras, lo que corrobora el resultado de la fase de entrenamiento. En la figura 5.4 se puede visualizar la matriz de confusión resultante, de forma que se tiene 901 casos de TN, 2176 de TP, 163 de FN y 336 de FP. Con estos valores se tiene que la sensibilidad es igual a 93.03 % y la Especifidad alcanza un valor del 72.84 %, de forma que se tiene un alto grado de fiabilidad en la clasificación de los datos.

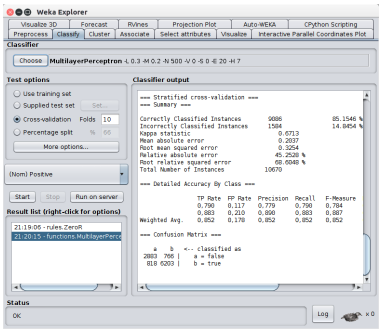
Los resultados dejan patente que la mínima energía libre de plegamiento, la energía de Gibbs, y el tipo de encaje de la región semilla son atributos que influyen notablemente en las interacciones entre microARN y ARNm (como se indicó en la introducción del documento). La evaluación realizada con la técnica Wrapper emplea para el análisis un clasificador SVM como base, pudiendo ser esta la causa de que se descartarán



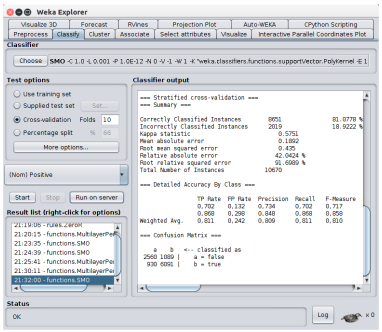
(a) Técnica: CFS con mejor el primero con red neuronal.



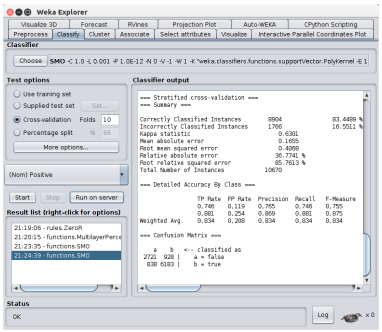
(b) Técnica: Relief con red neuronal.



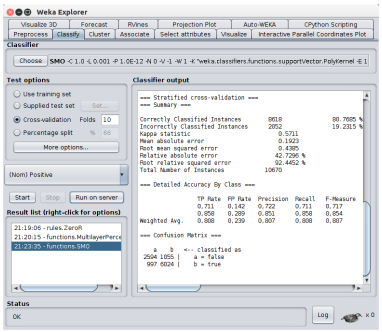
(c) Técnica: Wrapper con red neuronal.



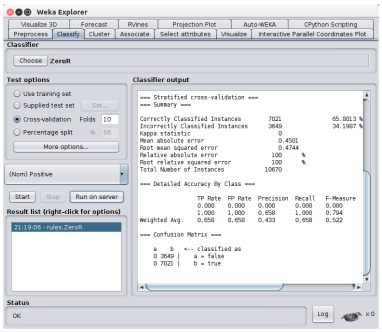
(d) Técnica: CFS con mejor el primero con SVM.



(e) Técnica: Relief con SVM.



(f) Técnica: Wrapper con SVM.



(g) Clasificador ZeroR.

Figura 5.2: Resultados del entrenamiento de los clasificadores.

estos 2 atributos. Asimismo, la técnica Relief, al ser un híbrido entre las dos grandes familias de técnicas FS, no descartó ningún parámetro, lo que dejó información redundante e innecesaria en los datos. Por otro lado, los SVM buscan hiperplanos que separen lo mejor posible la información que reciben, mientras que las redes proceden a calibrar el peso de las entradas en cada una de sus capas. Esta diferencia en el enfoque del problema puede ser la causa de que las redes neuronales generen mejores resultados.

5.3. Programa Java

La aplicación Java desarrollada se muestra en la figura 5.5. La interfaz está constituida por dos caja de texto. En la superior se ha de insertar la secuencia del miARN y en la inferior los nucleótidos que conforma el ARNm. Tras esto, al hacer click en el

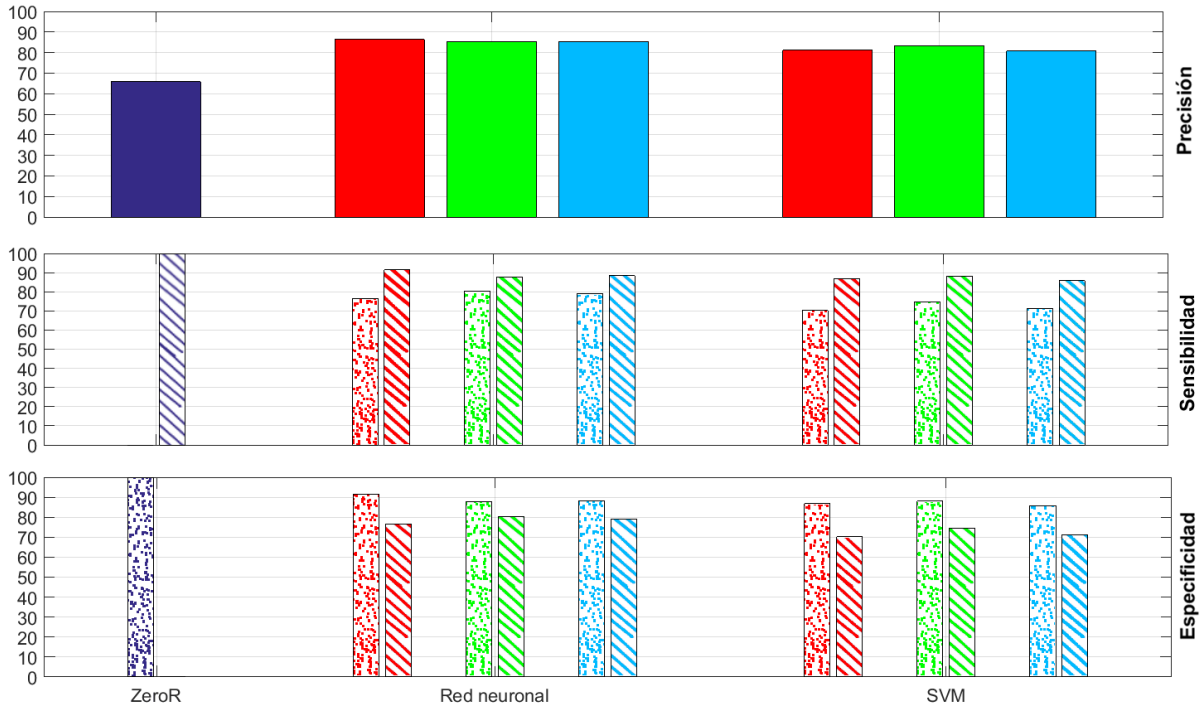


Figura 5.3: Resultado de los clasificadores en función del conjunto de atributos. El significado de los colores es el siguiente: morado hace referencia a la evaluación ZeroR, rojo identifica al resultado con 5 atributos que propuso el algoritmo CfsSubSetEva, verde engloba los resultados para todos los atributos seleccionados por RELIEF y azul a las 3 características que propuso la técnica Wrapper. Por otro lado, los puntos identifican a interacciones negativas, mientras líneas diagonales se asocian a las interacciones positivas.

botón *Play* se procede a buscar posibles dianas en el ARN aplicando la estrategia descrita en la sección 4.3.3. Con los segmentos, calculamos los parámetros de la interacción y se los pasamos como entrada al clasificador. Finalmente, si en alguno de los segmentos hay un caso positivo, entonces se muestra un mensaje informando ello, de lo contrario se le indicará al usuario que la interacción ha sido clasificada como negativa.

5.3.1. Requisitos

Para que la aplicación funcione correctamente es necesario tener instalado la máquina virtual Java en su versión 1.7 o superior así como el paquete de aplicaciones ViennaRNA 2.0. La aplicación ha sido probada con éxito en Linux y Windows, aunque en este último se exige que la ruta de instalación de Vienna sea *C:\Program Files (x86)\ViennaRNA Package*.

5.3.2. Estructura del proyecto

La estructura del proyecto se puede observar en la figura 5.5. Éste está configurado por 5 paquetes:

1. *database*: este paquete contiene 2 clases que son responsables de generar los datos para el proceso de entrenamiento y verificación de los clasificadores.

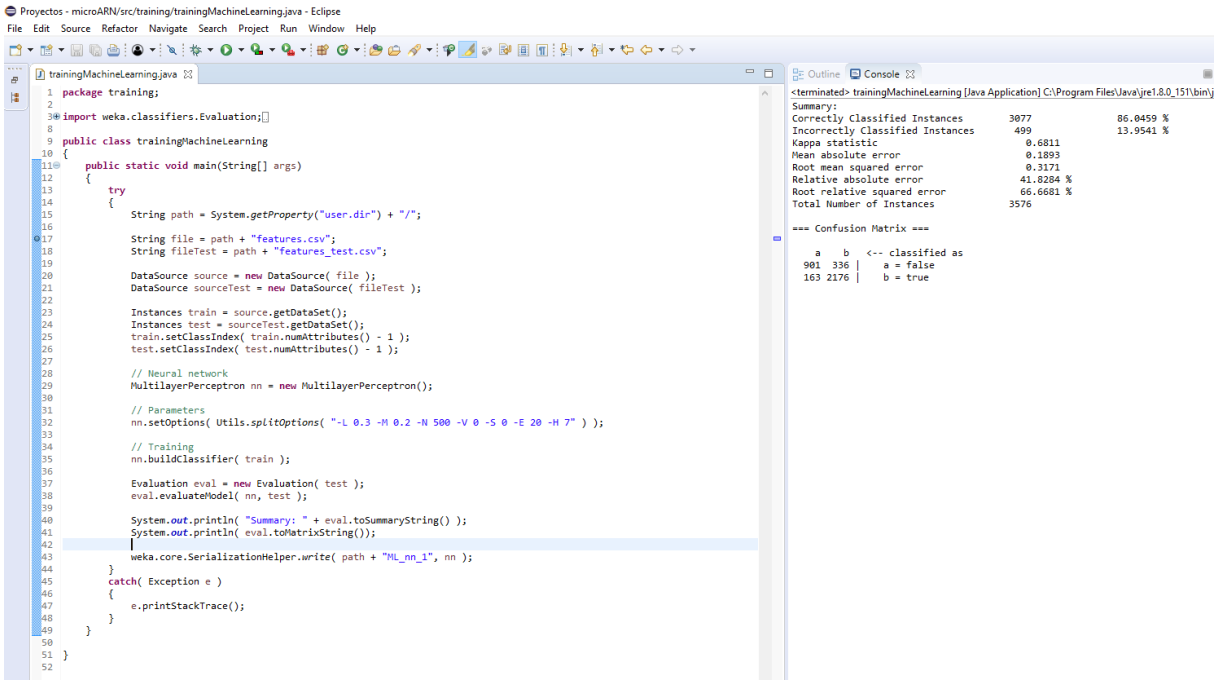


Figura 5.4: Evaluación de la precisión de la red neuronal al clasificar ejemplos no contenidos en el conjunto de entrenamiento.

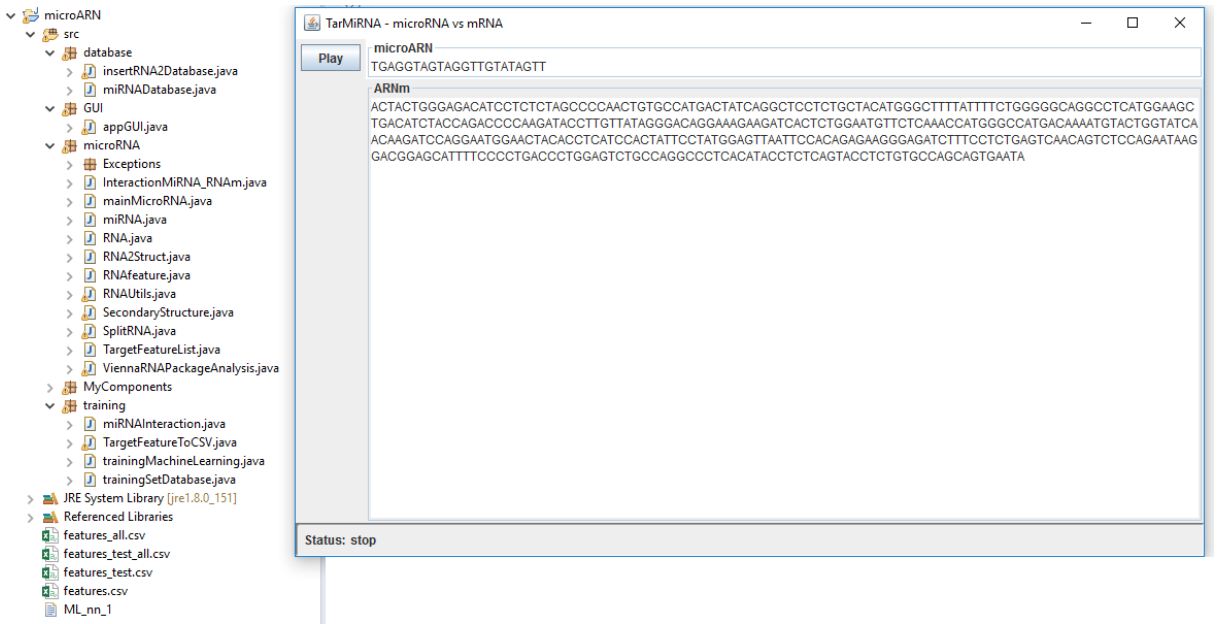


Figura 5.5: Interfaz gráfica del programa Java generado para la clasificación de posibles interacciones entre miARN y ARNmf.

- *miRNADatabase*: crea la conexión a la base de datos.
- *insertRNA2Database*: ejecutable que aplica la estrategia de generación de datos explicada en el capítulo 4.

2. *GUI*: paquete con el diseño y la lógica de ejecución para clasificar las entradas (figura 5.5).

3. *microRNA*: paquete con las clases necesarias para evaluar la interacciones miARN-ARNm.

   - *Exceptions*: subpaquete que recoge las clases que controlan las excepciones asociadas a las secuencias de nucleótidos. Son 2 clases: *NonRNASequenceException* que se da cuando los caracteres de la cadena no representan nucleótidos, y la otra es *SeedRegionException* se produce cuando la región del miARN no es correcta.
   - *InteractionMiRNA_RNA*: clase que se encarga de analizar la interacción resultante proporcionada por ViennaRNA. Ésta determina el tipo de encaje de semilla, porcentaje de puentes WC, porcentaje de uniones G-U, porcentaje de incompatibilidades entre nucleótidos, porcentaje de protuberancias tanto en el miARN como ARNm, etc.
   - *mainMicroRNA*: clase ejecutable que lanza la interfaz gráfica de la aplicación y controla si se produce alguna excepción en el proceso.
   - *miRNA*: representa al micro ARN. Almacena la secuencia completa y determina la semilla.
   - *RNA*: representa al ARNm. Contiene la secuencia completa y calcula las regiones UTRs (3' y 5').
   - *RNA2Struct*: La aplicación *RNAcofold* de ViennaRNA recibe como parámetro una única cadena de texto con las secuencias separadas por el signo & y devuelve otra secuencia de puntos y paréntesis donde el símbolo & divide la parte correspondiente a la primera de la segunda (figura 4.2). Esta clase se encarga de almacenar una secuencia de nucleótidos con su correspondiente alineamiento.
   - *RNAfeature*: clase que contiene el valor asociado a una única característica de las interacciones. Permite tanto parámetros numérico como cadenas de texto.
   - *RNAUtils*: conjunto de funciones útiles en el procesamiento de las interacciones. Permite convertir secuencias de ADN en ARN y al revés, descodificar las secuencias que contienen caracteres de la *Unión Internacional de Química Pura y Aplicada*, invertir una secuencia, determinar el tipo de unión entre nucleótido, verificar si la cadena de texto pasada como entrada es una secuencia correcta de nucleótidos, proporcionar el nucleótido complementario de otro y generar la secuencia complementaria.
   - *SecondaryStructure*: analiza el alineamiento generado por ViennaRNA. Calcula el porcentaje de uniones G-C, A-U y G-U, el número de protuberancias normalizadas entre 0 y 1 en el miARN y ARNm, el número de incompatibilidades entre secuencias normalizado entre 0 y 1, así como los porcentaje de los 8 tipos de emparejamientos de subsecuencias.

- *SplitRNA*: responsable del proceso de selección de subsecuencia candidata a ser diana del miARN en base a lo descrito en la sección 4.3.3.
- *TargetFeatureList*: clase que almacena los 26 parámetros que se extraen en el proceso de interacción.
- *ViennaRNaPackageAnalysis*: ejecuta el programa *RNAcofold* sobre las secuencias y obtiene los atributos a raíz del resultado.

4. *MyComponents*: paquetes con clases auxiliares que no influyen en el proceso de cálculo.

5. *training*: contiene los elementos responsables del entrenamiento del clasificador.

   - *miRNAInteraction*: almacena la información sobre las interacciones extraídas de la DB: identificador del miARN, identificador del gen y su transcriptor, la versión (segmento) de la subsecuencia del ARNm, número de la interacción y tipo de caso (positivo/negativo).
   - *TargetFeatureToCSV*: clase ejecutable que genera los archivos con los datos para entrenar y validar los clasificadores. Obtiene la información de la DB sobre las interacciones y secuencias, calcula los parámetros y los almacena en un fichero en formato CSV. Indicar que las secuencias son sometidas al criterio de rechazo que se indicó en las sección 4.3.3.
   - *trainingMachineLearning*: empleando la información generada en la clase *TargetFeatureToCSV* para entrenar y evaluar el clasificador de red neuronal que se seleccionó en la sección 5.2. Éste se almacena con el nombre *ML_nn_1* y debe estar en la misma ruta donde esté la aplicación Java para que no se dé ninguna excepción.
   - *trainingSetDatabase*: genera los conjuntos de entrenamientos y verificación en base a la numeración empleada en las interacciones que almacena la DB.

Capítulo 6

Discusión y conclusiones

El desarrollo de herramientas que permitan evaluar, con un alto grado de fiabilidad, información cuya estructura interna desconocemos abre un amplio abanico de posibilidades en el campo de la investigación, ya que facilitan saltarse un escollo y continuar indagando en otros elementos más interesantes. De esta forma, los algoritmos de machine learning y selección de características facilitan esta tarea en relación a la regulación de la expresión génica. No obstante, el desarrollo y aplicación de estos requieren de sumo cuidado a la hora de configurar los datos que se van a emplear el proceso. El uso de datos verificados por expertos es crucial a la hora de tener información veraz y fiable en la que basarse. En las interacciones entre miARN y ARNm es crucial usar datos que hayan sido previamente validados, tanto para generar las muestras positivas como las negativas. En este TFM se han usado ficheros públicos con información contrastada de las interacciones, y que a su vez, han servido tanto para generar casos positivos como negativos. Estos casos se han usados para generar un conjunto de atributos que se evaluaron y emplearon en el desarrollo de los clasificadores obteniendo un resultado final bastante aceptable (una precisión en la clasificación de los datos de entorno al 86 %). De esta forma, podemos concluir:

1. La búsqueda de información fiable es de suma relevancia a la hora de afrontar una labor de investigación y desarrollo. Emplear información de estudios previos nos evita reinventar la rueda, y poder avanzar más rápido y de forma más eficaz, además de proveernos de material e información útil (muestras de datos, parámetros a extraer, clasificadores empleados, etc.).

2. La generación de las muestras debe planificarse cuidadosamente y verificar que dicha información es correcta, ya que de lo contrario, el tiempo empleado para subsanar los errores se eleva notablemente, además de correrse el riesgo de que los resultados no sean fiables.

3. El uso de atributos sin emplear ninguna técnica de selección no siempre genera mejor resultado. De hecho, en este TFM se evaluaron el uso de todos los parámetros frente a solo 5 de ellos, de forma que en este último caso se obtuvieron mejores resultados. Más información no siempre es mejor, sino que puede emborronar los resultados. De esta forma, el uso de algoritmos de selección de características es recomendable.

4. Se aconseja evaluar diversas alternativas a la hora de clasificar la información. Cada técnica ML tiene su propio enfoque, de forma que puede resultar mejor

para resolver un problema frente otro clasificador.

5. Los resultados dejan patentes la importancia del tipo de encaje de la región semilla de los miARN y mínima energía libre de plegamiento en la precisión de los clasificadores, tal y como se apunto en la introducción del proyecto.

El objetivo que se planteó inicialmente en el proyecto era desarrollar una herramienta capaz de validar con un alto grado de fiabilidad si un miARN regula la expresión de un gen. Este objetivo se ha alcanzado a medias. Aunque se evaluaron 2 clasificadores sobre un conjunto de muestras amplio, el número de parámetros extraídos de éstos se vio limitado por problemas durante el desarrollo. Esto impidió que se evaluaran otros clasificadores. Asimismo, la generación de los casos negativos no se pudieron contrastar con fuentes validadas, lo que genera cierto grado de escepticismo en los resultados logrados.

La planificación establecida para el desarrollo del proyecto ha permitido cumplir sin demasiadas desviaciones los objetivos parciales marcados. Ésta se estructuro en 6 puntos: estudiar diferentes técnicas de ML y FS de forma que se tuviera cierto conocimiento a la hora de afrontar la búsqueda de bibliografía sobre micro ARN. Las referencias sobre predicción de interacciones entre ARN mensajero y miARN fue la base a la hora de determinar qué parámetros extraer y cómo calcularlos, qué algoritmos ML y FS se suelen emplear y de dónde extraer las muestras a usar. Finalmente, con la información anterior se generaron y evaluaron los clasificadores y se redactó el presente documento. Esta planificación fue pensada para que el desarrollo del proyecto fuera incremental, evitando posibles descuidos en el análisis de la información. No obstante, algunas de las fechas de los hitos estuvieron demasiado ajustadas al no prever desviaciones. Por ejemplo, la generación de la base de datos se retraso entorno a 1 semana respecto a la fecha planificada, lo que produjo que la tarea de entrenamiento se viera acortada notablemente. En cualquiera de los casos, el trabajo se finalizó a tiempo con un resultado aceptable.

Con todo esto, el trabajo presenta puntos de mejoras que se podrían ampliar en un futuro:

1. El trabajo se ha centrado en la especie *homo sapiens*, por lo que se podría ampliar a más organismos.

2. Buscar más muestras de casos positivos, y realizar una búsqueda más profundas de casos negativos validados.

3. Incorporar más atributos a los datos de entrenamiento y verificación.

4. Emplear de más técnicas de FS.

5. Evaluar más técnicas ML en la clasificación de las interacciones.

6. Mejorar la interfaz gráfica de la aplicación, de forma que además de informar de existe interacción, indique sobre qué subsecuencia de entrada.

Acrónimos

This document is incomplete. The external file associated with the glossary 'acronym' (which should be called `TFM_manuel_merino_monge.acr`) hasn't been created.

Check the contents of the file `TFM_manuel_merino_monge.acn`. If it's empty, that means you haven't indexed any of your entries in this glossary (using commands like `\gls` or `\glsadd`) so this list can't be generated. If the file isn't empty, the document build process hasn't been completed.

Try one of the following:

- Add `automake` to your package option list when you load `glossaries-extra.sty`. For example:

  ```
  \usepackage[automake]{glossaries-extra}
  ```

- Run the external (Lua) application:

  ```
  makeglossaries-lite "TFM_manuel_merino_monge"
  ```

- Run the external (Perl) application:

  ```
  makeglossaries "TFM_manuel_merino_monge"
  ```

Then rerun LaTeX on this document.
This message will be removed once the problem has been fixed.

Bibliografía

[Álvaro Ángel Orozco Gutiérrez, et al., 2005] Álvaro Ángel Orozco Gutiérrez, et al. (2005). Determinación de movimientos a partir de señales electromiográficas utilizando máquinas de soporte vectorial . *Revista Médica de Risaralda*, 11(c):15.

[Bahrammirzaee, 2010] Bahrammirzaee, A. (2010). A comparative survey of artificial intelligence applications in finance: Artificial neural networks, expert system and hybrid intelligent systems. *Neural Computing and Applications*, 19(8):1165–1195.

[Bandyopadhyay and Mitra, 2009] Bandyopadhyay, S. and Mitra, R. (2009). TargetMiner: MicroRNA target prediction with systematic identification of tissue-specific negative examples. *Bioinformatics*, 25(20):2625–2631.

[Barea et al., 2002] Barea, R., Boquete, L., Mazo, M., and López, E. (2002). Wheelchair guidance strategies using EOG. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 34(3):279–299.

[Bash, 2015] Bash, E. (2015). *Machine Learning with R*, volume 1.

[Batuwita and Palade, 2009] Batuwita, R. and Palade, V. (2009). microPred: Effective classification of pre-miRNAs for human miRNA gene prediction. *Bioinformatics*, 25(8):989–995.

[Bonnet et al., 2004] Bonnet, E., Wuyts, J., Rouzé, P., and Van de Peer, Y. (2004). Evidence that microRNA precursors, unlike other non-coding RNAs, have lower folding free energies than random sequences. *Bioinformatics*, 20(17):2911–2917.

[Cano et al., 2017] Cano, G., García-Rodríguez, J., Orts, S., García-García, A., Peña-García, J., Pérez-Garrido, A., and Pérez-Sánchez, H. (2017). Predicción de solubilidad de fármacos usando máquinas de soporte vectorial sobre unidades de procesamiento gráfico. *Revista Internacional de Metodos Numericos para Calculo y Diseno en Ingenieria*, 33(1-2):97–102.

[Cao and Tay, 2003] Cao, L. and Tay, F. E. H. (2003). Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transaction on Neural Networks*, 14(6):1506–18.

[Chandrashekar and Sahin, 2014] Chandrashekar, G. and Sahin, F. (2014). A survey on feature selection methods. *Computers and Electrical Engineering*, 40(1):16–28.

[Enright et al., 2003] Enright, A. J., John, B., Gaul, U., Tuschl, T., Sander, C., and Marks, D. S. (2003). MicroRNA targets in Drosophila. *Genome biology*, 5(1):R1.

[Frenger, 1999] Frenger, P. (1999). Linear circuits for neural networks and affective computing. In *Biomedical Sciences Instrumentation*, volume 35, pages 247–252.

[Friedman et al., 2009] Friedman, R. C., Farh, K. K. H., Burge, C. B., and Bartel, D. P. (2009). Most mammalian mRNAs are conserved targets of microRNAs. *Genome Research*, 19(1):92–105.

[Garcia-Martin and Clote, 2015] Garcia-Martin, J. A. and Clote, P. (2015). RNA thermodynamic structural entropy. *PLoS ONE*, 10(11).

[Grosswendt et al., 2014] Grosswendt, S., Filipchyk, A., Manzano, M., Klironomos, F., Schilling, M., Herzog, M., Gottwein, E., and Rajewsky, N. (2014). Unambiguous Identification of miRNA: Target site interactions by different types of ligation reactions. *Molecular Cell*, 54(6):1042–1054.

[Guyon and Elisseeff, 2003] Guyon, I. and Elisseeff, A. (2003). An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research (JMLR)*, 3(3):1157–1182.

[Guyon et al., 2006] Guyon, I., Gunn, S., Nikravesh, M., and Zadeh, L. (2006). *Feature Extraction, Foundations and Applications*, volume 207.

[Helwak et al., 2013] Helwak, A., Kudla, G., Dudnakova, T., and Tollervey, D. (2013). Mapping the human miRNA interactome by CLASH reveals frequent noncanonical binding. *Cell*, 153(3):654–665.

[Jevsinek Skok et al., 2013] Jevsinek Skok, D., Godnic, I., Zorc, M., Horvat, S., Dovc, P., Kovac, M., and Kunej, T. (2013). Genome-wide in silico screening for microRNA genetic variability in livestock species. *Animal Genetics*, 44(6):669–677.

[Kertesz et al., 2007] Kertesz, M., Iovino, N., Unnerstall, U., Gaul, U., and Segal, E. (2007). The role of site accessibility in microRNA target recognition. *Nature Genetics*, 39(10):1278–1284.

[Kim et al., 2016] Kim, D., Sung, Y. M., Park, J., Kim, S., Kim, J., Park, J., Ha, H., Bae, J. Y., Kim, S., and Baek, D. (2016). General rules for functional microRNA targeting. *Nature Genetics*, 48(12):1517–1526.

[Kim et al., 2006] Kim, S.-K., Nam, J.-W., Rhee, J.-K., Lee, W.-J., and Zhang, B.-T. (2006). miTarget: microRNA target gene prediction using a support vector machine. *BMC Bioinformatics*, 7:411.

[Lee et al., 1993] Lee, R. C., Feinbaum, R. L., and Ambros, V. (1993). The C. elegans heterochronic gene lin-4 encodes small RNAs with antisense complementarity to lin-14. *Cell*, 75(5):843–854.

[Lewis et al., 2005] Lewis, B. P., Burge, C. B., and Bartel, D. P. (2005). Conserved seed pairing, often flanked by adenosines, indicates that thousands of human genes are microRNA targets. *Cell*, 120(1):15–20.

[Lewis et al., 2003] Lewis, B. P., Shih, I. H., Jones-Rhoades, M. W., Bartel, D. P., and Burge, C. B. (2003). Prediction of Mammalian MicroRNA Targets. *Cell*, 115(7):787–798.

[Liu et al., 2010] Liu, H., Yue, D., Chen, Y., Gao, S.-J., and Huang, Y. (2010). Improving performance of mammalian microRNA target prediction. *BMC bioinformatics*, 11:476.

[Loher and Rigoutsos, 2012] Loher, P. and Rigoutsos, I. (2012). Interactive exploration of RNA22 microRNA target predictions. *Bioinformatics*, 28(24):3322–3323.

[Lopes et al., 2015] Lopes, I. D. O. N., Schliep, A., and de Carvalho, A. C. P. D. L. F. (2015). Automatic learning of pre-miRNAs from different species. *BMC Bioinformatics*, pages 1–30.

[Lorenz et al., 2011] Lorenz, R., Bernhart, S. H., Höner zu Siederdissen, C., Tafer, H., Flamm, C., Stadler, P. F., and Hofacker, I. L. (2011). ViennaRNA Package 2.0. *Algorithms for Molecular Biology*, 6(1).

[Lu and Leslie, 2016] Lu, Y. and Leslie, C. S. (2016). Learning to Predict miRNA-mRNA Interactions from AGO CLIP Sequencing and CLASH Data. *PLoS Computational Biology*, 12(7).

[Maragkakis et al., 2009] Maragkakis, M., Reczko, M., Simossis, V. A., Alexiou, P., Papadopoulos, G. L., Dalamagas, T., Giannopoulos, G., Goumas, G., Koukis, E., Kourtis, K., Vergoulis, T., Koziris, N., Sellis, T., Tsanakas, P., and Hatzigeorgiou, A. G. (2009). DIANA-microT web server: Elucidating microRNA functions through target prediction. *Nucleic Acids Research*, 37(SUPPL. 2).

[Mitchell, 1997] Mitchell, T. M. (1997). *Machine Learning*. Number 1.

[Pereira et al., 2007] Pereira, U. T. D., Andrés, G., España, M., Alexander, R., Cárdenas, B., and José, J. (2007). RECONOCIMIENTO DE COMANDOS POR VOZ CON MÁQUINAS DE SOPORTE VECTORIAL A TRAVÉS DE BANDAS ESPECTRALES. *Scientia et Technica*, XIII(37):79–84.

[Peterson et al., 2014] Peterson, S. M., Thompson, J. A., Ufkin, M. L., Sathyanarayana, P., Liaw, L., and Congdon, C. B. (2014). Common features of microRNA target prediction tools.

[Russell and Norvig, 2003] Russell, S. J. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition.

[Shardlow, 2016] Shardlow, M. (2016). An Analysis of Feature Selection Techniques. *The University of Manchester*, pages 1–7.

[Singh et al., 2010] Singh, R., Yadav, C. S., Verma, P., and Yadav, V. (2010). Optical Character Recognition (OCR) for Printed Devnagari Script Using Artificial Neural Network. *International Journal of Computer Science & Communication*, 1(1):91–95.

[Trotta, 2014] Trotta, E. (2014). On the normalization of the minimum free energy of RNAs by sequence length. *PLoS ONE*, 9(11).

[Vlachos et al., 2015] Vlachos, I. S., Paraskevopoulou, M. D., Karagkouni, D., Georgakilas, G., Vergoulis, T., Kanellos, I., Anastasopoulos, I. L., Maniou, S., Karathanou, K., Kalfakakou, D., Fevgas, A., Dalamagas, T., and Hatzigeorgiou, A. G. (2015). DIANA-TarBase v7.0: Indexing more than half a million experimentally supported miRNA:mRNA interactions. *Nucleic Acids Research*, 43(D1):D153–D159.

[Wass, 2007] Wass, J. A. (2007). Weka machine learning workbench. *Scientific Computing*, 24(3):21–47.

[Xiao et al., 2009] Xiao, F., Zuo, Z., Cai, G., Kang, S., Gao, X., and Li, T. (2009). miRecords: An integrated resource for microRNA-target interactions. *Nucleic Acids Research*, 37(SUPPL. 1).

[Yue et al., 2009] Yue, D., Liu, H., and Huang, Y. (2009). Survey of Computational Algorithms for MicroRNA Target Prediction. *Current genomics*, 10(7):478–92.

[Zuker, 1989] Zuker, M. (1989). On finding all suboptimal foldings of an RNA molecule. *Science*, 244(4900):48–52.

Anexo A

Parámetros descartados

Debido a diversos problema durante el desarrollo del proyecto, los siguientes parámetros han quedado sin incluir:

1. Características termodinámicas [Lopes et al., 2015, Yue et al., 2009]. Descartadas al no ser capaz de calcular dichos valores.
   a) Entalpía promedio por nucleótido.
   b) Entropía promedio por nucleótido [Garcia-Martin and Clote, 2015].
   c) Temperatura de fusión ($TM = \frac{\Delta S}{\Delta H}$.
   d) Energía de accesibilidad [Kertesz et al., 2007].
2. **Nivel de conservación**: de las secuencias entre las especies en las regiones UTR 3' y UTR 5'.
3. Longitud de la cadena sin codon de stop (TAA/TAG/TGA).
4. Porcentaje de pares de nucleótidos en el tallo del miARN: $\%(A+U)\_t$, $\%(G+C)\_t$ y $\%(G+U)\_t$. Descartado al no disponerse de esta información durante la generación de los datos.
5. Mínima energía libre de plegamiento normalizada [Trotta, 2014]. Descartado por falta de tiempo.
6. Energía libre de banda normalizada. Descartado por falta de tiempo.
7. Diferencia normalizada de energía libre ($\frac{MFE-EFE}{L}$). Descartado por falta de tiempo.
8. Frecuencia de la estructura MFE. Descartado al no ser capaz de calcularlo.
9. Entropía de Shannon normalizada. Descartado al no ser capaz de calcularlo.
10. Diversidad estructural. Descartado al no ser capaz de calcularlo.
11. Porcentaje de las regiones de complejidad baja. Descartado al no ser capaz de calcularlo.