

UNIVERSIDAD NACIONAL DEL LITORAL



DOCTORADO EN INGENIERÍA

Nuevo enfoque de aprendizaje semi-supervisado para la identificación de secuencias en bioinformática

Cristian Ariel Yones

FICH
FACULTAD DE INGENIERÍA
Y CIENCIAS HÍDRICAS

sinc(i)
INSTITUTO DE INVESTIGACIÓN EN SEÑALES
SISTEMAS E INTELIGENCIA COMPUTACIONAL

INTEC
INSTITUTO DE DESARROLLO TECNOLÓGICO
PARA LA INDUSTRIA QUÍMICA

CIMEC
CENTRO DE INVESTIGACIÓN DE
MÉTODOS COMPUTACIONALES



UNIVERSIDAD NACIONAL DEL LITORAL
Facultad de Ingeniería y Ciencias Hídricas
Instituto de Investigación en Señales, Sistemas e Inteligencia Computacional

**NUEVO ENFOQUE DE APRENDIZAJE
SEMI-SUPERVISADO PARA
LA IDENTIFICACIÓN DE
SECUENCIAS EN BIOINFORMÁTICA**

Cristian Ariel Yones

Tesis remitida al Comité Académico del Doctorado como
parte de los requisitos para la obtención del grado de
DOCTOR EN INGENIERÍA
Mención en Inteligencia Computacional, Señales y Sistemas
de la
UNIVERSIDAD NACIONAL DEL LITORAL

2018



UNIVERSIDAD NACIONAL DEL LITORAL
Facultad de Ingeniería y Ciencias Hídricas
Instituto de Investigación en Señales, Sistemas e Inteligencia Computacional

**NUEVO ENFOQUE DE APRENDIZAJE
SEMI-SUPERVISADO PARA
LA IDENTIFICACIÓN DE
SECUENCIAS EN BIOINFORMÁTICA**

Cristian Ariel Yones

Lugar de Trabajo:
sinc(i)

Instituto de Señales, Sistemas e Inteligencia Computacional
Facultad de Ingeniería y Ciencias Hídricas
Universidad Nacional del Litoral

Director:
Dr. Diego H. Milone *sinc(i)*, CONICET-UNL

Co-director:
Dra. Georgina Stegmayer *sinc(i)*, CONICET-UNL

Jurado Evaluador:

DECLARACIÓN LEGAL DEL AUTOR

Esta Tesis ha sido remitida como parte de los requisitos para la obtención del grado académico de Doctor en Ingeniería ante la Universidad Nacional del Litoral y ha sido depositada en la Biblioteca de la Facultad de Ingeniería y Ciencias Hídricas para que esté a disposición de sus lectores bajo las condiciones estipuladas por el reglamento de la mencionada Biblioteca.

Se permiten citaciones breves de esta Tesis sin la necesidad de un permiso especial, en la suposición de que la fuente sea correctamente citada. El portador legal del derecho concedera por escrito solicitudes de permiso para la citación extendida o para la reproducción parcial o total de ese manuscrito serán concebidos por el portador legal del derecho de propiedad intelectual de la obra, por medio escrito.

TESIS POR COMPILACIÓN

La presente tesis se encuentra organizada bajo el formato de Tesis por Compilación, aprobado en la resolución No 255/17 (Expte. No 888317-17) por el Comité Académico de la Carrera Doctorado en Ingeniería, Facultad de Ingeniería y Ciencias Hídricas, Universidad Nacional del Litoral (UNL). De dicha resolución:

“En el caso de optar por la Tesis por Compilación, ésta consistirá en una descripción técnica de al menos 30 páginas, redactada en español e incluyendo todas las investigaciones abordadas en la tesis. Se deberán incluir las secciones habituales indicadas a continuación en la Sección Contenidos de la Tesis. Los artículos científicos publicados por el autor, en el idioma original de las publicaciones, deberán incluirse en un Anexo con el formato unificado al estilo general de la Tesis indicado en la Sección Formato. El Anexo deberá estar encabezado por una sección donde el tesista detalle para cada una de las publicaciones cuál ha sido su contribución. Esta sección deberá estar avalada por su director de Tesis. El documento central de la Tesis debe incluir referencias explícitas a todas las publicaciones anexadas y presentar una conclusión que muestre la coherencia de dichos trabajos con el hilo conceptual y metodológico de la tesis. Los artículos presentados en los anexos podrán ser artículos publicados, aceptados para publicación (en prensa) o en revisión.”

Índice general

1. Introducción	1
1.1. Aprendizaje automático semi-supervisado	1
1.2. Predicción automática de microARN	2
1.3. Objetivo general	4
1.4. Objetivos específicos	4
2. Métodos propuestos	6
2.1. Procesamiento de secuencias de ARN de tipo tallo-horquilla	6
2.1.1. Ventaneo del genoma	6
2.1.2. Plegado y poda	6
2.1.3. Recorte de secuencias	7
2.1.4. Filtrado de repetidas	7
2.2. Extracción de características	8
2.2.1. Secuencia primaria	9
2.2.2. Estructura secundaria	9
2.2.3. Estabilidad termodinámica	10
2.2.4. Estabilidad estadística	10
2.2.5. Conservación filogenética	11
2.2.6. Análisis de subcadenas de 22 nt	11
2.3. Clasificación de pre-miARNs	12
2.3.1. Construcción del grafo	13
2.3.2. Búsqueda de ejemplos negativos	13
2.3.3. Estimación de puntajes de predicción	14
2.3.4. Umbralización de los puntajes de predicción	16
3. Resultados	17
3.1. Procesamiento de secuencias de ARN de tipo tallo-horquilla	17
3.2. Predicción de pre-microARN	17
4. Conclusiones	24
5. Publicaciones	26
Apéndices	32
Contribuciones	33
HextractoR: an R package for automatic extraction of hairpins from genome-wide data	34
miRNAfe: a comprehensive tool for feature extraction in microRNA prediction	40
Genome-wide pre-miRNA discovery from few labeled examples	66

Índice de tablas

3.1. Cantidad de horquillas y pre-miARN en varios genomas	18
3.2. Comparación de tiempos de ejecución	19
3.3. Resultados en genoma completo	23

Índice de figuras

1.1.	Aprendizaje semi-supervisado vs supervisado	2
1.2.	Aprendizaje inductivo vs. transductivo	3
1.3.	Estructura secundaria de un pre-miARN	4
2.1.	Etapas de la predicción de microARN	7
2.2.	Extracción de secuencias tipo tallo-horquilla	8
2.3.	Evolución del grafo	12
3.1.	Sensibilidad en animales y plantas	19
3.2.	\bar{G} con pocos ejemplos de entrenamiento	20
3.3.	<i>AUC</i> con pocos ejemplos positivos	21
3.4.	Curvas ROC en genoma completo	22

Resumen

El aprendizaje maquinal ha tenido un gran desarrollo en los últimos años y ha permitido resolver una gran cantidad de problemas en las más diversas disciplinas. Sin embargo, aún quedan grandes desafíos por resolver, como lo es el aprendizaje en datos con alto grado de desbalance de clases o con muy pocos datos etiquetados. Un caso particular de aplicación donde se presentan desafíos como estos es en la predicción computacional de secuencias de microARN (miARN). Los microARN (miARN) son un grupo de pequeñas secuencias de ácido ribonucleico (ARN) no codificante que desempeñan un papel muy importante en la regulación génica. En los últimos años, se han desarrollado una gran cantidad de métodos que intentan detectar nuevos miARNs utilizando sólo información de estructura y secuencia, es decir, sin medir niveles de expresión. El primer paso en estos métodos generalmente consiste en extraer del genoma subcadenas de nucleótidos que cumplan con ciertos requerimientos estructurales. En segundo lugar se extraen características numéricas de estas subcadenas para finalmente usar aprendizaje maquinal para predecir cuáles probablemente contengan miARN. Por otro lado, en paralelo con los métodos de predicción de miARN se han propuesto una gran cantidad de características para representar numéricamente las subcadenas de ARN. Finalmente, la mayoría de los métodos actuales usan aprendizaje supervisado para la etapa de predicción. Este tipo de métodos tienen importantes limitaciones prácticas cuando deben aplicarse a tareas de predicción real. Existe el desafío de lidiar con un número escaso de ejemplos de pre-miARN positivos. Además, es muy difícil construir un buen conjunto de ejemplos negativos para representar el espectro completo de secuencias no miARN. Por otro lado, en cualquier genoma, existe un enorme desequilibrio de clase (1 : 10000) que es bien conocido por afectar particularmente a los clasificadores supervisados.

Para permitir predicciones precisas y rápidas de nuevos miARNs en genomas completos, en esta tesis se realizaron aportas en las tres etapas del proceso de predicción de miARN. En primer lugar, se desarrolló una herramienta para extraer subcadenas de un genoma completo que cumplan con los requerimientos mínimos para ser potenciales pre-miARNs miARN. En segundo lugar, se desarrolló una herramienta que permite calcular la mayoría de las características utilizadas para predicciones de miARN en el estado del arte. La tercera y principal contribución consiste en un algoritmo novedoso de aprendizaje semi-supervisado que permite realizar predicciones a partir de muy pocos ejemplos de clase positiva y el resto de las cadenas sin etiqueta de clase. Este tipo de aprendizaje aprovecha la información provista por las subcadenas desconocidas (sobre las que se desea generar predicciones) para mejorar las tasas de predicción. Esta información extra permite atenuar el efecto del número reducido de ejemplos etiquetados y la pobre representatividad de las clases. Cada herramienta diseñada fue comparada contra el estado del arte, obteniendo mejores tasas de desempeño y menores tiempos de ejecución.

Sección 1

Introducción

1.1. Aprendizaje automático semi-supervisado

El aprendizaje automático es un campo de las ciencias de la computación que intenta dotar a los sistemas informáticos de la capacidad de aumentar progresivamente su habilidad en una tarea específica, sin la necesidad de ser específicamente programados para tal tarea (Samuel, 1959). Dentro de este campo podemos identificar dos grandes grupos de técnicas: las de aprendizaje supervisado y las de aprendizaje no supervisado. Las primeras utilizan un conjunto de datos que traen asociados una etiqueta de clase (que indica a qué clase pertenece el objeto) o valor de salida deseado. El proceso de aprendizaje consiste en generar un modelo que logre mapear los datos de entrada a los valores de salida deseados (Russell and Norvig, 2016). El modelo además debe ser capaz de generalizar, es decir, generar salidas correctas cuando reciba como entrada datos no vistos durante el proceso de aprendizaje. Por otro lado, en el aprendizaje no supervisado el objetivo es generar modelos que descubran grupos o relaciones subyacentes en los datos sin utilizar ningún tipo de etiqueta o valor de salida esperado.

Como punto intermedio entre los dos grupos de técnicas antes mencionados, se ha propuesto un nuevo grupo de algoritmos, llamados de aprendizaje semi-supervisado (Chapelle *et al.*, 2006). Este tipo de algoritmos utiliza ejemplos etiquetados para aprender la distribución de las clases de la misma forma que los métodos de aprendizaje supervisado, pero además aprovecha ejemplos no etiquetados para modelar mejor el espacio de las características y así mejorar las tasas de clasificación. Como los ejemplos no etiquetados proveen información extra sobre la distribución de los datos en el espacio de las características, este grupo de técnicas obtiene buenas tasas de clasificación en problemas donde la cantidad de datos etiquetados es baja o no es representativa de su clase. En la Figura 1.1 podemos ver un ejemplo muy simple de cómo datos sin etiqueta pueden ayudar a reconocer los límites de las clases de interés. En la Figura 1.1.a vemos 4 ejemplos de entrenamiento de dos clases diferentes y la frontera de decisión que construiría un clasificador supervisado con estos datos de entrada. En el siguiente cuadro vemos que los datos de entrenamiento en realidad no eran representativos de sus clases por lo que la frontera de decisión falla al separarlas. Por otro lado, si se utiliza un esquema semi-supervisado (Figura 1.1.c) este aprovecha tanto los datos etiquetados como los no etiquetados para descubrir la distribución latente de los datos y de esta forma separar las dos clases satisfactoriamente.

Un concepto que está estrechamente relacionado con el aprendizaje semi-supervisado es el de aprendizaje transductivo. En esta configuración, los datos de entrenamiento se utilizan para hacer directamente inferencias en datos de prueba. A diferencia de la mayoría de los métodos inductivos, donde el algoritmo tiene dos etapas (entrenamiento e inferencia), los métodos transductivos tienen sólo una etapa. En el esquema inductivo (Figura 1.2.a) el algoritmo de aprendizaje utiliza datos de entrenamiento etiquetados para ajustar un modelo. En la segunda etapa, el modelo ya ajustado se usa para hacer predicciones sobre datos nuevos (no vistos durante el entrenamiento). En el esquema transductivo (Figura 1.2.b) tanto los datos etiquetados como los no etiquetados se procesan juntos, en la misma etapa. Como resultado, se obtienen etiquetas para los datos no etiquetados, pero ninguna función de decisión, modelo entrenado o regla de clasificación. Cabe señalar que la validación de cualquier método de transducción es bastante diferente de una estándar (inductiva). Dado que el entrenamiento y la

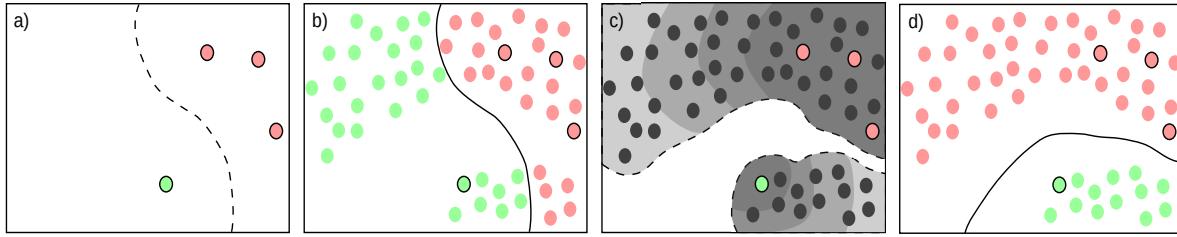


Figura 1.1: Ejemplo de clasificación con aprendizaje supervisado (a-b) y con aprendizaje semi-supervisado (c-d).

predicción se combinan en una etapa, al algoritmo se le proporcionan tanto los datos de entrenamiento como los de prueba, pero los segundos sin etiquetas. Este tipo especial de validación se utiliza en todos los algoritmos transductivos (Chapelle *et al.*, 2006), donde las etiquetas de los casos de prueba nunca se utilizan en ningún paso de la etapa de predicción.

1.2. Predicción automática de microARN

En los últimos años, la biología molecular demostró un importante avance en distintas áreas de la investigación. Esta disciplina científica estudia la estructura, función y composición de las moléculas biológicamente importantes y se relaciona fuertemente con otras áreas como la genética y la bioquímica. En la actualidad se invierte un gran esfuerzo en estudiar las interacciones entre los numerosos sistemas de las células, incluyendo las interacciones entre los diferentes tipos de ácido desoxirribonucleico (ADN), ácido ribonucleico (ARN) y los procesos de síntesis de proteínas, y también cómo estas interacciones son reguladas. Los avances en este campo permiten la generación de una gran cantidad de información que requiere el uso de herramientas de cálculo altamente especializadas para el análisis. En este contexto, la bioinformática ha tomado un papel muy importante ya que aporta herramientas que posibilitan la explotación de estos datos.

Un problema abierto en Bioinformática desde hace más de una década es el de la predicción automática de microARNs (miARNs). Los miARNs son una clase de pequeñas moléculas (~ 21 nucleótidos) monocatenarias de ácido desoxirribonucleico no codificante que regulan la expresión de otros genes. Se encuentran presentes tanto en animales como en plantas y su importancia en procesos biológicos clave ha sido ampliamente documentada (Rosenzvit *et al.*, 2013). Los miARNs están implicados, por ejemplo, en la evolución del cáncer (sea como inhibidores o promotores de éste) (Yu *et al.*, 2015), o en procesos de infección viral (Lecellier *et al.*, 2005). Además, identificar los miARNs de ciertas especies tiene aplicaciones directas, como puede ser la mejora de cultivos (Liu and Chen, 2010) o, en animales, el desarrollo de nuevas vacunas y antibióticos (Tsetsarkin *et al.*, 2017). Estas pequeñas moléculas normalmente se transcriben a partir de secuencias primarias de mayor tamaño, llamadas pri-miARNs. Durante la biogénesis, las secuencias precursoras se pliegan sobre sí mismas formando una estructura secundaria tipo tallo-horquilla como se puede ver en la Figura 1.3. Luego son separadas de la secuencia principal y trasladadas fuera del núcleo. En este paso, la estructura toma el nombre de precursora del miARN o pre-miARN. Finalmente el miARN es liberado del precursor para tomar un papel activo en la regulación de la expresión de los genes.

En la última década, se utilizaron numerosos enfoques experimentales y computacionales para identificar nuevos miARNs. Los métodos experimentales que se han usado tradicionalmente como el clonado directo son capaces de detectar sólo miARNs que se expresan de forma abundante (Kleftogiannis *et al.*, 2013). Los últimos avances en secuenciación permiten un alto nivel de cobertura, mitigando estos efectos (An *et al.*, 2013). Sin embargo, aquellos que se expresan en ciertas etapas de una especie, o en tejidos específicos, pueden no ser detectados. Por otro lado, el desarrollo y posterior análisis de los modelos computacionales entrenados para reconocer miARNs puede brindar pistas que permitan caracterizar mejor estas moléculas y su biogénesis. Los enfoques computacionales utilizados para la clasificación de secuencias candidatas a miARN pueden dividirse en dos grupos: métodos por homología y métodos de aprendizaje automático. Los primeros se basan en la conservación de miARNs entre

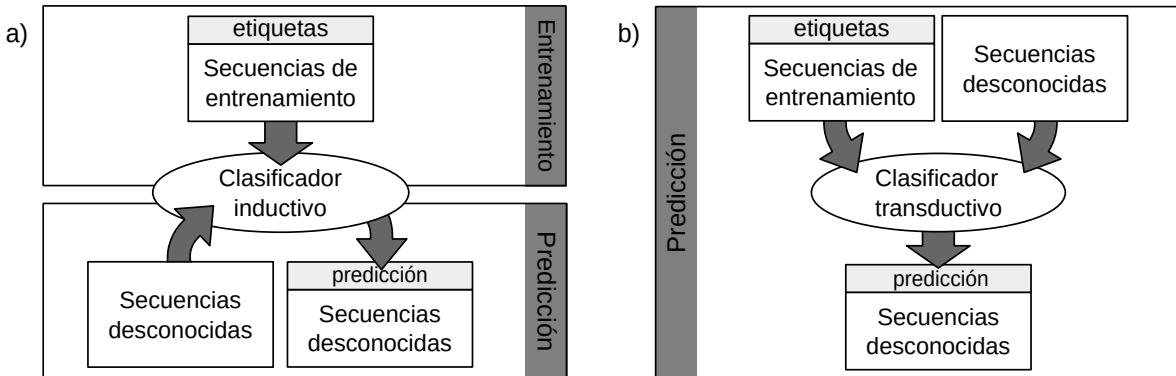


Figura 1.2: Aprendizaje inductivo y transductivo. a) Esquema tradicional inductivo con dos etapas separadas; b) Esquema transductivo con sólo una etapa.

especies estrechamente relacionadas y, por lo tanto, no pueden usarse para miARNs que son específicos de una especie (Ng and Mishra, 2007). Los métodos de aprendizaje automático utilizan un enfoque diferente, que consiste en extraer primero características de pre-miARNs conocidos, de secuencias que se utilizarán como ejemplos negativos y de las secuencias desconocidas que se desea clasificar. Después se utilizan las características de las secuencias conocidas (tanto positivas como negativas) para entrenar un clasificador de forma supervisada, para finalmente aplicarlo a las secuencias no conocidas y obtener una predicción (Kleftogiannis *et al.*, 2013). Las características que pueden usarse para representar secuencias de ARN han sido ampliamente estudiadas (Lopes *et al.*, 2014). Además, muchos algoritmos de aprendizaje supervisado han sido aplicados a este problema: máquinas de soporte vectorial, modelos ocultos de Markov y ensambles de clasificadores, entre otros (Kleftogiannis *et al.*, 2013). Sin embargo todavía existen muchos problemas abiertos en la predicción de pre-miARNs con técnicas de aprendizaje maquinal (Stegmayer *et al.*, 2018).

La gran mayoría de los métodos de predicción del estado del arte toman como entrada secuencias con estructura secundaria tipo tallo-horquilla y se encargan luego de extraer características, entrenar un clasificador y hacer predicciones. Si bien este primer paso generalmente es ignorado, es fundamental para lograr buenos resultados en las etapas posteriores. No sólo es importante lograr capturar todas las secuencias con estructura secundaria tipo tallo-horquilla del genoma, sino que también es fundamental que estas secuencias se corten en los lugares correctos. Si un pre-miARN se corta erróneamente con una longitud mayor o menor a la real, las características calculadas pueden variar mucho y generar resultados erróneos durante la etapa de predicción. Actualmente, la única herramienta disponible para esta tarea (Durbin and Rice, 1999) no aprovecha los últimos avances en cuanto a la estimación de estructuras secundarias, por lo que el número de secuencias que detecta es bajo y muchos pre-miARN se pierden durante esta temprana etapa. Además, no se tiene en cuenta el contexto de la secuencia para detectar dónde son los mejores puntos de corte. Las características utilizadas en la predicción de pre-miARN son muy sensibles a pequeñas diferencias en el corte, por lo que elegir el punto óptimo es crucial para obtener buenos resultados en la clasificación.

Por otro lado, durante la última década se ha desarrollado una gran cantidad de características específicamente para el problema de predicción de microARN, que se pueden extraer tanto de las secuencias o sus estructuras secundarias. Se han publicado muchas herramientas para calcular algunas de estas características, pero estas están codificadas en diferentes lenguajes de programación y tienen diferentes modos de acceso (web, línea de comandos, etc.). Además, varias de estas herramientas son software propietario y el código fuente no está disponible¹. Estas cuestiones dificultan en gran medida el uso de muchas de estas características.

Por último, la etapa de predicción o clasificación de secuencias como posibles candidatos a pre-miARN aún presenta importantes dificultades que no han sido resueltas satisfactoriamente. En primer lugar, aunque el número de secuencias no pre-miARN que se pueden encontrar en cualquier genoma

¹<http://www.insybio.com/pages/ncrnaseq>



Figura 1.3: Estructura secundaria de un pre-miARN humano. En rojo se puede ver resaltado el miARN hsa-let-7e.

es grande, los ejemplos negativos que se utilicen deben ser representativos de la gran variedad de secuencias no pre-miARN. En (Wei *et al.*, 2014) se analizó la importancia de los ejemplos negativos utilizados y cómo esto limita el rendimiento de los clasificadores. En la mayoría de los casos, los conjuntos negativos que se usan para ajustar los modelos de clasificación se definen principalmente con secuencias tomadas de regiones de codificación de proteínas, ARNm u otras regiones donde es poco probable encontrar miARN (Peace *et al.*, 2015; Tempel *et al.*, 2015). Pero nada puede asegurar que estas secuencias sean buenos representantes de todas las posibles secuencias no pre-miARN, o que estén lo suficientemente cerca de los límites de la verdadera clase de pre-miARN. Por ejemplo, si el poder de predicción de un clasificador se prueba en un conjunto de datos que tiene pre-miARNs y ncARNs, lo que realmente se está midiendo es la capacidad de diferenciar entre estos dos tipos de secuencia, pero el clasificador podría no descartar correctamente otras secuencias no pre-miARN. Ningún método ha informado tasas de error teniendo en cuenta este punto crucial. Para abordar este problema, algunos métodos utilizan secuencias tomadas de posiciones aleatorias de un genoma como ejemplos negativos (Wenyuan *et al.*, 2013; Gudýš *et al.*, 2013). Sin embargo, en este caso, nada puede garantizar que un pre-miARN no conocido se utilice erróneamente como un ejemplo negativo. En ambas estrategias de construcción de conjuntos de ejemplos negativos, no se puede garantizar que los ejemplos sean representativos de toda la clase negativa; y, lo que es peor, algunos de ellos podrían ser falsos negativos. Otra característica de este problema que dificulta la aplicación simple de técnicas de aprendizaje maquinal es el hecho de que el número de ejemplos positivos es muy bajo en cualquier genoma, por lo cual este constituye un problema con muy alto desbalance entre clases. Este problema es aún peor en especies no modelo, donde la cantidad de secuencias anotadas es muy baja. Una alternativa puede ser utilizar pre-miARNs de especies cercanas como ejemplos positivos, pero recientemente se ha demostrado (Lopes *et al.*, 2016) que el problema de predicción de pre-miARNs es muy dependiente de la especie, por lo que esta estrategia puede afectar el desempeño de las técnicas de predicción. Por otro lado, la gran variedad de secuencias que se pueden utilizar como ejemplos negativos y el bajo número de ejemplos positivos obliga a entrenar clasificadores con un desbalance muy grande entre el número de secuencias positivas y las potencialmente negativas. Esto presenta una dificultad adicional ya que está demostrado que las técnicas estándar de aprendizaje maquinal son muy sensibles al desbalance de clases (Guo *et al.*, 2008).

1.3. Objetivo general

El objetivo general de esta tesis es desarrollar nuevos métodos para la clasificación de candidatos a pre-miARN a partir del genoma completo tanto de animales como de plantas, y capaces de desempeñarse adecuadamente en un contexto de gran desbalance entre la clase de interés y una alta proporción de secuencias no etiquetadas.

1.4. Objetivos específicos

A continuación se detallan los objetivos específicos de la presente investigación:

- Desarrollar una metodología integrada y simple que permita encontrar automáticamente secuen-

cias tipo tallo-horquilla en genomas completos.

- Revisar e integrar todas las características utilizadas en la actualidad para la predicción de pre-miARNs.
- Desarrollar nuevos algoritmos de clasificación que incorporen etapas de aprendizaje semi-supervisado para atacar los problemas de gran desbalance de clases y alta proporción de ejemplos no etiquetados.
- Validar las propuestas con genomas reales a través del trabajo multidisciplinario con expertos del dominio.

Sección 2

Métodos propuestos

Siguiendo las ideas presentadas previamente, en la Figura 2 se presenta la metodología general diseñada para la predicción de nuevos microARNs. El proceso consiste en tres etapas, en primer lugar se deben extraer todas las subcadenas del genoma que tengan una estructura secundaria similar a la de un pre-miARN. Luego, a estas secuencias candidatas se les extraen características para convertirlas en vectores numéricos, procesables con métodos de aprendizaje maquinal. En paralelo, se comparan todas las subcadenas con los pre-miARNs conocidos para marcar los ejemplos positivos. Finalmente, se obtienen predicciones para las secuencias no etiquetadas utilizando un algoritmo de aprendizaje maquinal semi-supervisado. En las siguientes subsecciones se describe con más detalle cada etapa del proceso.

2.1. Procesamiento de secuencias de ARN de tipo tallo-horquilla

Como se mencionó antes, los pre-miARNs se pliegan en estructuras secundarias tipo tallo-horquilla. Estas secuencias tienen una longitud promedio que depende de la especie, y un nivel de emparejamiento relativamente alto. Estas características especiales de los pre-miARNs, además de ser útiles para una primer etapa de filtrado, se pueden utilizar para cortar el genoma completo en subcadenas que pueden ser procesadas más fácilmente por las siguientes etapas. El objetivo de esta etapa es entonces extraer todas las secuencias tipo tallo-horquilla de un genoma cortando en los puntos donde se obtienen las estructuras secundarias con la mayor estabilidad termodinámica posible.

Para ello, se predice la estructura secundaria de varios segmentos superpuestos de una longitud más larga que la longitud media de las secuencias de la especies de interés, asegurándose de que nada se pierda ni se corte de manera inapropiada. La longitud de la ventana de corte se puede configurar para definir el tamaño máximo que tendrán las secuencias encontradas. Los pasos de procesamiento (ver Figura 2.1) se explican en detalle en las siguientes secciones.

2.1.1. Ventaneo del genoma

Se comienza cortando el genoma completo en ventanas solapadas de una longitud varias veces mayor (500 nt.) a la de un pre-miARN promedio. La ventana debe ser lo suficientemente larga para poder capturar la horquilla completa pero además para que se tenga en cuenta el entorno de cualquier posible horquilla al estimar la estructura secundaria. Esto último es muy importante ya que los resultados de estimar la estructura secundaria se ven muy afectados por el entorno de la secuencia. Por ejemplo, sin considerar el entorno de una secuencia, la estructura secundaria estimada puede presentar una estabilidad menor a la que se formaría si se habría considerado el entorno.

2.1.2. Plegado y poda

Se predice la estructura secundaria que formaría cada una de las ventanas al plegarse. Para ello se utiliza el algoritmo de mínima energía libre (Zuker and Stiegler, 1981). Este algoritmo utiliza pro-

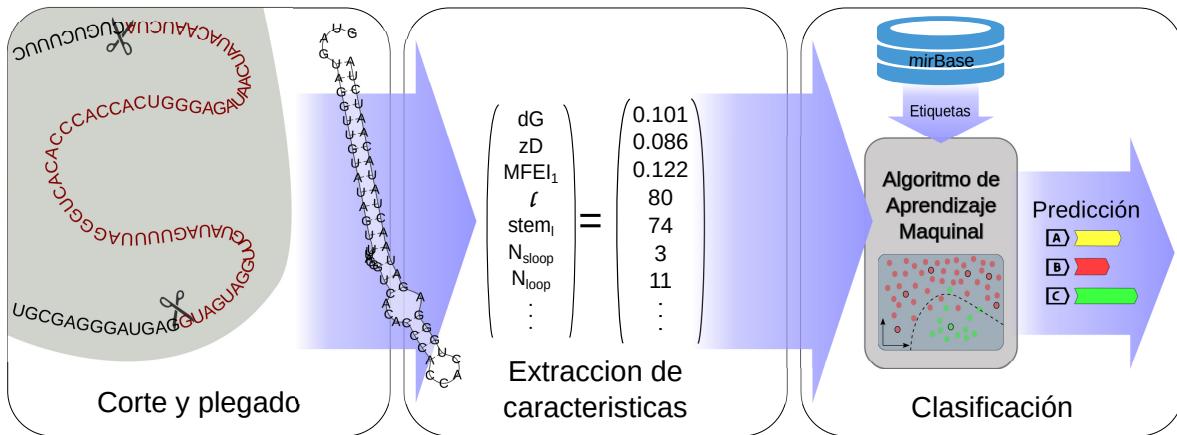


Figura 2.1: Proceso completo de predicción de nuevos pre-miARNs, separado en sus tres etapas principales.

gramación dinámica para encontrar la estructura secundaria que minimiza la energía liberada. Dado que las ventanas utilizadas son relativamente largas, las estructuras normalmente presentan múltiples bucles. Por lo tanto, el siguiente paso es buscar los puntos de corte que logren separar la estructura completa en varias estructuras tipo horquilla, como se puede ver en la Figura 2.2. De las horquillas extraídas, se eliminan las que no superan una longitud y nivel de emparejamiento mínimo y el resto pasa a la siguiente etapa.

2.1.3. Recorte de secuencias

Como se utilizan ventanas de una longitud varias veces mayor a la de un pre-miARN promedio, generalmente las secuencias encontradas forman estructuras secundarias mayores a la de un pre-miARN. Por este motivo es importante analizar cada secuencia para detectar si es necesario un recorte. Como aún se desconoce el funcionamiento de las enzimas que se encargan de separar el pre-miARN de la secuencia, se utilizan ciertas heurísticas que permiten obtener secuencias con longitudes y estabilidad similares a las de un pre-miARN. Estas reglas optimizan la energía libre mínima normalizada por la longitud de la secuencia (NMFE, por sus siglas en inglés *Normalized Minimum Free Energy*). Aunque se pueden encontrar puntos de corte óptimos re-estimando la estructura secundaria para todos los posibles cortes, utilizar reglas heurísticas proporciona más flexibilidad y acelera el proceso. Primero, la secuencia debe exceder una longitud mínima, predefinida según la especie en estudio. De esta manera, se puede asegurar que la estructura secundaria tenga una longitud suficiente para ser un pre-miARN de la especie en análisis. En segundo lugar, los cortes se realizan en el primer nucleótido no emparejado de un bucle interno o bullo de la estructura secundaria (comenzando desde el bucle principal). Para elegir el bucle o bullo donde cortar se asigna una puntuación a cada imperfección como $S(1 - D/L)^2$, donde S es la longitud de la imperfección, D es la distancia al bucle principal y L es la longitud del tallo (desde el lazo principal hasta el último nucleótido emparejado). Si las imperfecciones en la estructura secundaria son grandes, es probable que cortar la secuencia en esos puntos genere una estructura con menor NMFE. Por otra parte, cuanto menor es la longitud de la secuencia (independientemente del nivel de emparejamiento), mayor es la NMFE. Por lo tanto, se favorecen los bucles o bultos más cercanos al bucle principal.

2.1.4. Filtrado de repetidas

Las secuencias repetidas se eliminan para evitar costo computacional extra en las siguientes etapas. Estas secuencias repetidas además perturban los resultados de los algoritmos de predicción utilizados en la última etapa, ya que cada repetición aumenta la “importancia” que se le da a una secuencia. Dado que la ventana se mueve de forma solapada, es posible que una estructura secundaria sea capturada más

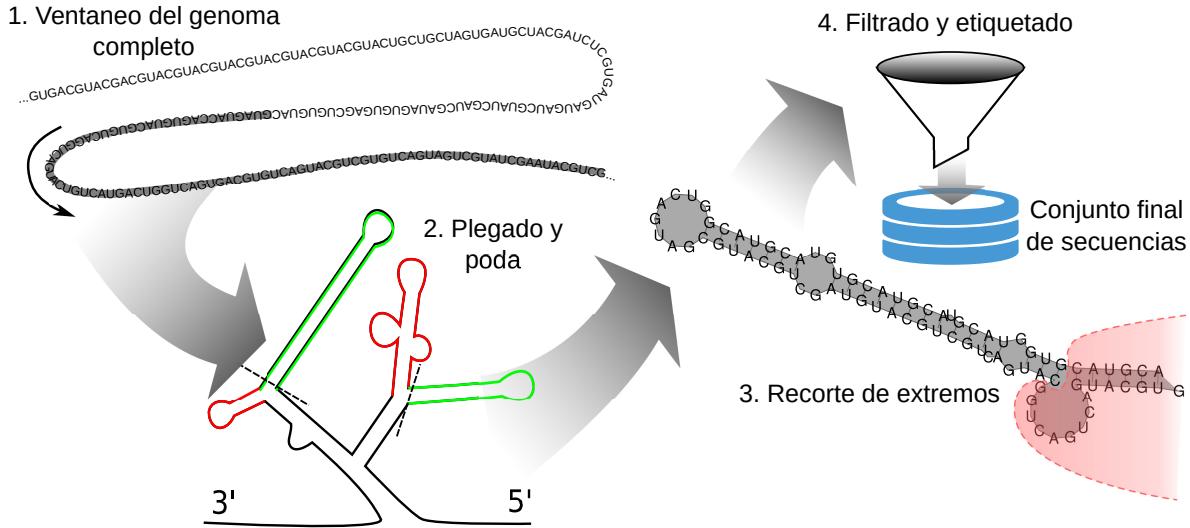


Figura 2.2: Proceso de extracción de secuencias tipo tallo-horquilla del genoma completo.

de una vez. Estas repeticiones aparecen de forma consecutiva y son secuencias casi idénticas. Entonces, lo único que puede variar es la longitud, dado que una horquilla puede caer en uno de los extremos de la ventana, generando una horquilla más corta. Para eliminarlas se realiza una comparación de cada secuencia con las últimas secuencias extraídas. Si una de las secuencias contiene a la otra, la más corta se elimina.

2.2. Extracción de características

La extracción de características de pre-miARN consiste en convertir las secuencias de nucleótidos (representadas como cadenas de caracteres) en vectores numéricos. De esta manera se pueden aplicar técnicas de aprendizaje maquinal para lograr separar las secuencias pre-miARN de las que no lo son. Como antes se expresó, se han presentado muchas características que pueden usarse para representar secuencias de ARN (Lopes *et al.*, 2014), pero la variedad de herramientas para extraerlas son un problema que dificulta su utilización. Estas tienen diferentes interfaces de acceso (web, línea de comandos, interfaz gráfica, etc.) y no siempre son de libre acceso. Para resolver este problema se creó una biblioteca que implementa de forma unificada todos los procesos de extracción publicados en la literatura en los últimos 10 años. Esta herramienta puede extraer características de la mayoría de los métodos de predicción de miARN de los últimos años: Triplet-SVM (Xue *et al.*, 2005), RNAmicro (Hertel and Stadler, 2006), BayesMiRNAfind (Yousef *et al.*, 2006), MiRFinder (Huang *et al.*, 2007), MiPred (Jiang *et al.*, 2007), miRRim (Terai *et al.*, 2007), microPred (Batuwita and Palade, 2009), miRanalyzer (Hackenberg *et al.*, 2009), MiRenSVM (Ding *et al.*, 2010) y miPredGA (Xuan *et al.*, 2011a).

La biblioteca, llamada miRNAfe, se construyó de forma modular para facilitar el proceso de agregar nuevos algoritmos de extracción de características en el futuro. Además de la biblioteca, se creó una interfaz web para simplificar el acceso a usuarios del dominio de aplicación sin conocimientos de programación. Tanto la biblioteca como la versión web pueden calcular hasta 80 características donde muchas son matrices multidimensionales. Las características se han dividido en seis grupos predefinidos: secuencia primaria, estructura secundaria, estabilidad termodinámica, estabilidad estadística, conservación entre genomas de diferentes especies y análisis de subcadenas de las secuencias. En las próximas secciones se detalla en qué consiste cada grupo de características. Una lista completa de las características descritas con mayor detalle se presentan en el Anexo 5.

2.2.1. Secuencia primaria

Estas son las características más simples y representan información de la secuencia principal. MiRNAfe puede extraer un total de 5 características en este grupo: longitud de la secuencia (ℓ), proporción de cada base en la secuencia, proporción de dinucleótidos, contenido de guanina y citosina y relación guanina-citosina. Las dos últimas características se definen como:

$$G + C_{content} = \frac{G + C}{G + C + A + U}, \quad (2.1)$$

$$GC_{ratio} = \frac{G}{C}, \quad (2.2)$$

donde G , C , A y U representan la cantidad de cada base encontrada en la secuencia (Hertel and Stadler, 2006). Todas estas características forman un vector de 23 elementos, compuesto por: las 4 proporciones base, las 16 proporciones de dinucleótidos, la longitud de la secuencia, $G + C_{contenido}$ y GC_{ratio} . Aunque estas características son bastante simples, han mostrado un alto poder discriminatorio (Batuwita and Palade, 2009), y por lo tanto se usan en la mayoría de herramientas de predicción.

2.2.2. Estructura secundaria

Estas características representan información de la estructura secundaria y son el grupo más numeroso. La característica más utilizada de este grupo, quizás por ser una de las primeras publicadas para la predicción de pre-miARN, es la proporción de tripletas (Xue *et al.*, 2005). Una tripleta es un elemento formado con el estado de estructura (emparejado o no emparejado) de tres nucleótidos y el tipo de base del nucleótido del medio. Un ejemplo de una tripleta es “. ((A”, donde el paréntesis representa un nucleótido emparejado, un punto uno no emparejado, y la letra es la base del nucleótido del medio. Como hay 2 estados posibles para un nucleótido y 4 bases diferentes, se pueden formar 32 tripletas (4×2^3). El número de ocurrencias de cada elemento en la secuencia se cuenta y se normaliza para producir un vector de 32 características. Un enfoque similar a las tripletas fue utilizado por Huang *et al.* (2007), que propuso otra representación para la estructura secundaria. En primer lugar, se definen cinco símbolos para indicar el estado de cada par de bases en el tallo: ‘=’, ‘:’, ‘-’, ‘.’ y ‘^’. Cada uno de ellos corresponde al estado de coincidencia, falta de coincidencia, eliminación, inserción por bucle interior e inserción por bullo, respectivamente. Luego, al tomar dos símbolos adyacentes, se pueden formar 14 posibles combinaciones, cada una de las cuales tiene un significado especial. Por ejemplo: ‘= -’, ‘= .’, y ‘= :’ representan el límite del bucle principal, y ‘: ^’ representa que el bucle es asimétrico. La frecuencia de cada combinación se usa como un vector de características. Esta representación también se puede usar para calcular cuatro características: *pMatch*, *pMismatch*, *pDI* y *pBulge* (Huang *et al.*, 2007). Estas características se calculan sobre supuestos miARN maduros, seleccionados como la región de 22 nucleótidos donde el apareamiento de bases es máximo. Representan la frecuencia de emparejamiento de bases, la frecuencia de no emparejamiento, las frecuencias de borrado e inserción y la simetría de los bucles abombados, respectivamente.

Otro tipo de características se relaciona con estructuras llamadas tallos, que son motivos estructurales que contienen más de tres pares de bases contiguas (Ng and Mishra, 2007). Estas características son el número de tallos, la proporción de cada posible par de bases por tallo, el número promedio de pares de bases por tallo y la longitud del tallo más largo. El resto de las características son la longitud de la región del tallo (es decir, la cantidad de nucleótidos de la horquilla sin contar los que se encuentran en el bucle principal, que no se debe confundir con los llamados tallos en Ng and Mishra (2007)), longitud del bucle terminal, número de bucles, número de bultos, longitud de bucle más largo, número de bucles simétricos y asimétricos, nucleótidos en bucles simétricos y asimétricos, región simétrica más larga, longitud promedio de bucles simétricos, longitud promedio de bucles asimétricos, cantidad de bultos y bucles de longitud 1, 2, ..., 6 y mayor que 7, número de pares de bases, proporción de pares de bases, proporción de pares de bases ajustada y $G + C_{contenido}$ en el bucle terminal (Lopes *et al.*, 2014). Finalmente, la biblioteca permite calcular el recuento de lecturas a partir de los datos de RNAseq. Esta característica necesita que el usuario proporcione un archivo adicional con lecturas, que se alinea con las secuencias analizadas y cuenta sus correspondencias.

2.2.3. Estabilidad termodinámica

La característica más utilizada de este grupo es la energía mínima libre (*MFE*, por sus siglas en inglés *Minimum Free Energy*): la energía estimada que una secuencia libera cuando se pliega en la estructura secundaria más estable (Zuker and Stiegler, 1981). La energía libre del conjunto (*EFE*) tiene un significado similar y se obtiene con el algoritmo de McCaskill (1990). Otras características de este grupo se calculan como combinaciones de esos valores. Por ejemplo, el índice *MFE* 1 (*MFEI*₁) es la relación entre la energía libre mínima y el *G + C*_{contenido} definido en (2.1). Del mismo modo, se puede calcular la diferencia *MFE – EFE*, *MFE* ajustado, *MFEI*₂, *MFEI*₃ y *MFEI*₄ (Batuwita and Palade, 2009). También hay algunas características que utilizan enfoques teóricos de información para estimar la confianza de la estructura secundaria predicha, como la entropía ajustada de Shannon de las probabilidades de emparejamiento (Ng and Mishra, 2007), definida como

$$dQ = \frac{1}{\ell} \sum_{i < j} p_{ij} \log_2 p_{ij}, \quad (2.3)$$

donde p_{ij} es la probabilidad de que el nucleótido i forme un par con el nucleótido j y ℓ es la longitud de la secuencia. Las probabilidades de cada par se calculan con el algoritmo de McCaskill (1990). Otro ejemplo es la distancia de pares ajustada, definida como

$$dD = \frac{1}{\ell} \sum_{i < j} p_{ij}(1 - p_{ij}). \quad (2.4)$$

Además, en este grupo se puede calcular la frecuencia del conjunto, la diversidad, el potencial del tallo 3' y 5', y el potencial del bucle (Terai *et al.*, 2007). Hay 15 funciones en este grupo, que se describen con más detalle en el Anexo 5.

2.2.4. Estabilidad estadística

Se sabe que los precursores que contienen un miARN son más estables que las secuencias aleatorias. Las características de este grupo se calculan como la puntuación estándar (z-score) de cualquier característica relacionada con la estabilidad (Bonnet *et al.*, 2004). Para calcular este valor, se debe generar una población aleatoria de secuencias intercambiando las bases de la secuencia analizada. De esta forma, las secuencias generadas artificialmente conservan las proporciones de nucleótidos o incluso la proporción de dinucleótidos si algunos intercambios están restringidos (la herramienta tiene una opción para elegir qué método de intercambio usar). Utilizando las secuencias generadas, la estabilidad de la secuencia original se puede medir con

$$z = \frac{x - \mu}{\sigma}, \quad (2.5)$$

donde x es el valor original de alguna característica relacionada con la estabilidad, μ es la media y σ es la desviación estándar de la población de secuencias generada aleatoriamente. Este puntaje representa, para un valor, cuántas desviaciones estándar un valor está por encima de la media de la población. Por lo tanto, un puntaje z negativo indica una secuencia que es estadísticamente más estable que la media de la población. Otra estadística utilizada para medir la estabilidad de la secuencia en comparación con secuencias aleatorias es el valor p . Se calcula como la proporción de secuencias aleatorias que son más estables que la secuencia analizada. Por lo tanto, un bajo valor de p indica que la secuencia analizada es una de las secuencias más estables generadas con esa proporción de nucleótidos (o dinucleótidos). Las medidas de estabilidad que se pueden normalizar con z-score son:

- *MFE* (*zMFE*);
- *EFE* (*zEFE*);
- *MFE* ajustado (*zG*);
- entropía de Shannon (*zQ*);

- propensión de pares de bases (zP) (Ng and Mishra, 2007) y
- distancia de pares de bases (zD) (Ding *et al.*, 2010).

El valor p se puede usar para normalizar MFE ($pMFE$) (Bonnet *et al.*, 2004) y EFE ($pEFE$) (Ding *et al.*, 2010). Aunque el z-score y el p -value son similares, a menudo se usan juntas en predicción ya que pueden tomar valores muy diferentes (Ding *et al.*, 2010). En resumen, se pueden calcular 8 características en este grupo. Además se puede especificar el método de mezcla (preservación de la composición de nucleótidos o dinucleótidos) y el número de secuencias aleatorias generadas.

2.2.5. Conservación filogenética

Cuando una parte del genoma se conserva entre especies relacionadas, es muy probable que tenga un papel importante en el genoma. Las características de este grupo miden el nivel de conservación entre secuencias de especies relacionadas filogenéticamente. Todas las características se calculan sobre alineaciones de dos o más secuencias que el usuario debe proporcionar. Algunas características no sólo tienen en cuenta el nivel de conservación, sino también la estabilidad termodinámica. Las características de ese grupo son:

- frecuencia de mutación (Huang *et al.*, 2007), que es la proporción de bases que difieren de una secuencia a otra;
- entropía del brazo 5', el brazo 3', la región del bucle y la entropía mínima (la menor entropía de las calculadas sobre una región de 21 nucleótidos (Hertel and Stadler, 2006));
- número de diferencias en la estructura secundaria dividido por el número de diferencias entre las secuencias (Huang *et al.*, 2007);
- promedio de MFE , dG y $MFEI_1$;
- diferencia de MFE entre dos secuencias alineadas dividida por el número de diferencias entre las secuencias (Huang *et al.*, 2007);
- energía libre de la estructura secundaria de consenso;
- conservación del brazo 3' y del brazo 5' y
- puntaje de conservación (Terai *et al.*, 2007), que se calcula usando dos procesos de Markov, uno que se mueve en la dimensión de tiempo (sobre las ramas del árbol de evolución) y el otro en dimensión espacial (sobre la secuencia);

Se pueden extraer un total de 14 características en este grupo, que se describen en detalle en el Anexo 5.

2.2.6. Análisis de subcadenas de 22 nt

Estas características se calculan en todas las subcadenas de 22 nt dentro de una secuencia determinada. Se basan en el hecho de que si una secuencia es un pre-miARN, una de las subcadenas analizadas tiene que ser el miARN maduro y las características calculadas deben capturar sus particularidades. Como resultado, se obtiene una matriz con longitud $n = \ell - 22$, donde el elemento i -th representa el valor de la característica calculada sobre la subcadena que comienza en la base i . MiRNAfe puede extraer las siguientes 5 características en este grupo:

- probabilidad de emparejamiento de bases en la subcadena (Lim *et al.*, 2003), que es la suma de probabilidades de emparejamiento de bases sobre la subcadena;
- suma de bases no emparejadas en la subcadena;
- suma de probabilidades de emparejamiento de bases en la estructura secundaria sin las probabilidades de los nucleótidos de la subcadena;

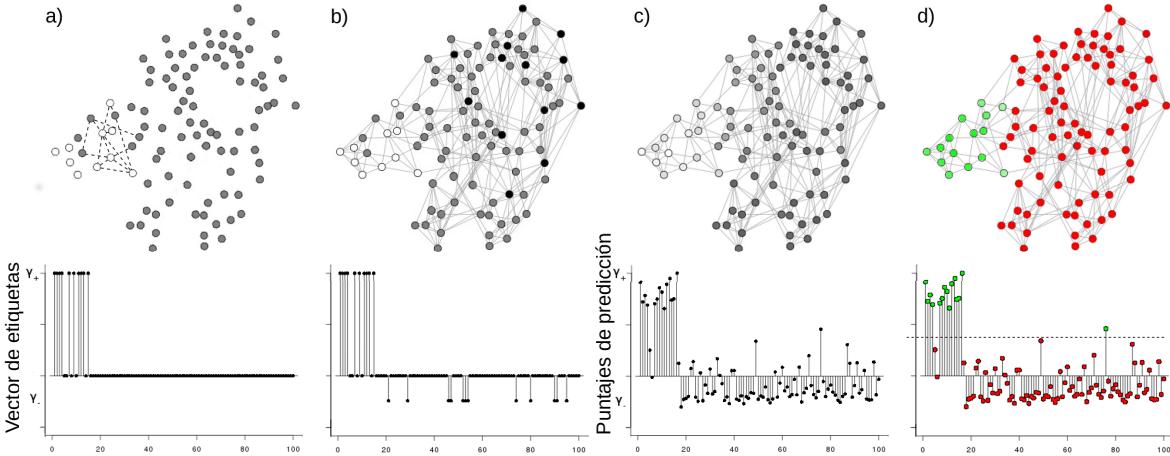


Figura 2.3: Evolución simplificada del grafo, el vector de etiquetas y el vector de puntajes de predicción en los 4 pasos del método propuesto: a) Construcción de grafo; b) inicialización de conjunto negativo; c) Estimación de puntajes de predicción; d) optimización del umbral.

- simetría de bultos, como la diferencia entre la cantidad de bases no emparejadas en cada brazo de la subcadena;
- distancia desde la subcadena hasta el bucle terminal.

2.3. Clasificación de pre-miARNs

En esta etapa se utilizan pre-miARNs conocidos del genoma de la especie de interés para entrenar un clasificador que realizará predicciones sobre otras secuencias de ARN, sobre las cuales no se conoce su función. Las secuencias de pre-miARN utilizadas como ejemplos para el entrenamiento pueden ser tomadas de alguna base de datos de pre-miARNs como mirBase (Kozomara and Griffiths-Jones, 2014), mientras que las secuencias no etiquetadas son el resultado del corte y plegado del genoma crudo. Como antes se mencionó, utilizar métodos de aprendizaje maquinal supervisado presenta numerosas limitaciones. Como enfoque más novedoso y superador del problema, se desarrolló un clasificador semi-supervisado que aprenda la distribución de secuencias del genoma analizado en el espacio de las características, utilizando las secuencias no etiquetadas. Aprovechando la información que aportan estas secuencias, se pretende mejorar las tasas de predicción en situaciones donde la cantidad de ejemplos es baja o cuando no son representativos de sus respectivas clases.

Durante la última década, una de las áreas más activas en el aprendizaje semi-supervisado ha sido la de los métodos basados en grafos, donde cada nodo representa un punto de datos, y un borde conecta nodos si son de alguna manera similares. Luego, utilizando los nodos etiquetados, se obtienen etiquetas pronosticadas para el resto de los nodos sin etiqueta. Estos métodos han mostrado buenas tasas de predicción (Joachims *et al.*, 2003) y han podido manejar grandes volúmenes de datos. Se ha desarrollado un nuevo algoritmo de predicción de este tipo que tiene en cuenta las características particulares del problema: a) grandes volúmenes de datos; b) desbalance de clases muy alto; c) clase negativa mal representada o directamente ausencia de ejemplos negativos. A continuación se describe el nuevo clasificador propuesto, denominado miRNAss.

MiRNAss recibe como entrada un conjunto de vectores de características m -dimensional \mathbf{x}_i , que representan secuencias, y un vector correspondiente de etiquetas ℓ . El elemento i -th en el vector de etiquetas tiene un valor positivo γ_+ si la secuencia i -th es un pre-miARN conocido, un valor negativo γ_- si no es un pre-miARN, y cero si es una secuencia desconocida que debe ser clasificada por el método. El método tiene cuatro pasos: 1) construcción de un grafo donde cada nodo representa una secuencia; 2) búsqueda de ejemplos negativos, si no se proporcionan; 3) estimación de los puntajes de

predicción para cada nodo en el grafo; y 4) estimación de un umbral óptimo para separar las secuencias en dos clases.

La Figura 2.3 muestra un ejemplo de la aplicación del método. En este ejemplo, 15 secuencias son verdaderos pre-miARN y el resto son secuencias no pre-miARN. En la Figura 2.3.a, el grafo se construye y se asignan los valores iniciales a ℓ : valores positivos (γ_+) a los ejemplos conocidos de pre-miARNs. En la Figura 2.3.b, algunos nodos que están topológicamente lejos de los ejemplos positivos se etiquetan como ejemplos negativos (γ_-). Los nodos del grafo están coloreados de acuerdo con los valores en el vector ℓ . En la Figura 2.3.c, los puntajes de predicción se estiman para todas las secuencias, teniendo en cuenta que: i) tienen que ser suaves; es decir, las secuencias topológicamente cercanas en el grafo deben tener puntajes de predicción similares; y ii) los puntajes tienen que ser similares a los valores distintos de cero dados en el vector de etiqueta ℓ . Finalmente, usando los puntajes de predicción asignados a los ejemplos marcados se estima un umbral óptimo para separar los pre-miARN de las otras secuencias. Las secuencias que pasan el umbral están coloreadas en verde; el resto, en rojo.

2.3.1. Construcción del grafo

Al construir el grafo, cada nodo representa una secuencia y cada arista une dos nodos si las secuencias son similares en el espacio de las características. Para medir la similaridad se utiliza la distancia euclídea entre los vectores de características. Si una característica no ayuda realmente a discriminar entre clases, puede empeorar el rendimiento del clasificador. Por lo tanto, es importante preprocessar los vectores de características para ponderar adecuadamente cada característica de acuerdo con su poder de predicción. Un algoritmo que ha demostrado mejorar los resultados en clasificadores que son sensibles a la función de distancia es el algoritmo RELIEF-F (Kononenko, 1994; Wettschereck *et al.*, 1997). Además, es computacionalmente eficiente y puede usarse para grandes volúmenes de datos. Funciona de la siguiente manera: comenzando con un vector de pesos con m ceros, para cada ejemplo busca el ejemplo más cercano de la misma clase y el ejemplo más cercano que pertenezca a la otra clase. Luego, aumenta los pesos de las características que son similares al ejemplo de la misma clase y diferentes al ejemplo de la otra clase. Por el contrario, se reducen los pesos de las características que son diferentes al ejemplo de la misma clase o similares al ejemplo de la otra clase. El resultado se llama vector de relevancia y tiene valores altos para las características más discriminativas. Si la relevancia de una característica es negativa, entonces no ayuda a discriminar entre clases y por lo tanto puede ser eliminada. El resto de las características se escala por su puntaje de relevancia para dar más peso a las más discriminativas.

Para la construcción del grafo, una opción común es el algoritmo de k -vecinos más cercanos (KNN), dado que es simple, rápido, eficiente en uso de memoria y fácilmente paralelizable. KNN construye una matriz de adyacencia ponderada por similitud con elementos

$$a_{ij} = \begin{cases} \frac{\mu}{\mu + ||\mathbf{x}_i - \mathbf{x}_j||^2} & \text{si } \mathbf{x}_j \in \mathcal{K}(\mathbf{x}_i) \text{ y } \ell_i \ell_j \geq 0 \\ 0 & \text{en otro caso,} \end{cases} \quad (2.6)$$

donde \mathbf{x}_i es el vector de características correspondiente a la secuencia i -th, $\mathcal{K}(\mathbf{x}_i)$ es el conjunto de los k vecinos más cercanos de \mathbf{x}_i , y μ es la media de las distancias entre las secuencias conectadas y se utiliza para normalizar los pesos de las aristas.

2.3.2. Búsqueda de ejemplos negativos

Si sólo se cuenta con ejemplos positivos, la matriz de adyacencia se puede utilizar para etiquetar un conjunto de secuencias como ejemplos negativos. La idea clave es seleccionar aleatoriamente algunas secuencias sin etiqueta entre las más distantes a los ejemplos positivos. De esta forma, la probabilidad de etiquetar incorrectamente un pre-miARN bien conocido como un ejemplo negativo será baja. Para ello, se calcula una medida de la similitud de cada secuencia con los ejemplos positivos utilizando la distancia topológica. Este vector de similitud, llamado s , se inicializa con +1 en los elementos correspondientes a ejemplos positivos. Los nodos sin etiqueta se inicializan con 0. Entonces, se puede usar un método iterativo para actualizar las similitudes en s (ver Algoritmo 1). En cada iteración y para cada nodo, el valor de similitud correspondiente de cada vecino se multiplica por los pesos

Algoritmo 1 Búsqueda automática de ejemplos negativos

Entrada: matriz de adyacencia A , vector de etiquetas ℓ con ceros y valores positivos únicamente, número de ejemplos negativos a etiquetar T .

Salida: vector ℓ con T etiquetas negativas asignadas.

- 1: $s_i = \begin{cases} 1 & \text{si } \ell_i > 0 \\ 0 & \text{en otro caso} \end{cases}$
- 2: **repetir**
- 3: $s_i = \max_{\forall j \neq i} \{s_i, a_{ij}s_j\}, \quad \forall i$
- 4: **mientras** no hay cambios en s
- 5: $p_i = e^{1-s_i} - 1, \quad \forall i$
- 6: Muestrear T elementos de ℓ usando p como pesos para la selección
- 7: Etiquetar elementos elegidos como clase negativa
- 8: **devolver** ℓ

de aristas correspondientes que los conectan. El valor máximo de los resultados obtenidos para cada vecino se compara luego con el valor de similitud actual del nodo. Si es más alto, se actualiza el valor de similitud del nodo. Cuando no hay más cambios en s , se seleccionan T secuencias de forma aleatoria usando como probabilidad de selección $p_i = e^{1-s_i} - 1$ y se etiquetan como ejemplos negativos.

2.3.3. Estimación de puntajes de predicción

En el tercer paso, los puntajes de predicción se calculan resolviendo un problema de optimización (Joachims *et al.*, 2003). Como se dijo anteriormente, se deben considerar dos puntos: i) los puntajes de predicción deben ser topológicamente suaves; y ii) las predicciones deben ser similares a las etiquetas conocidas. Para suavizar la predicción se minimiza el cuadrado de las diferencias entre las puntuaciones de predicción de las secuencias adyacentes. Una representación conveniente para calcular fácilmente estas diferencias es el laplaciano normalizado del grafo (Shi and Malik, 2000), definido como

$$L = I - D^{-1/2}AD^{-1/2}$$

donde

$$D_{ij} = \begin{cases} \sum_{k=0}^n A_{ik} & \text{si } i = j \\ 0 & \text{en otro caso.} \end{cases}$$

El Laplaciano tiene una propiedad útil para medir la suavidad de la solución. Supongamos $\mathbf{z} \in \mathbb{R}^N$, con una predicción para cada nodo del gráfico. Entonces,

$$\begin{aligned} \mathbf{z}^T L \mathbf{z} &= \mathbf{z}^T I \mathbf{z} - \mathbf{z}^T D^{-1/2} A D^{-1/2} \mathbf{z} = \\ &= \sum_i^n z_i^2 - \sum_i^n \sum_j^n \frac{z_i}{\sqrt{d_{ii}}} \frac{z_j}{\sqrt{d_{jj}}} a_{ij} = \\ &= \frac{1}{2} \sum_i^n \sum_j^n a_{ij} \left(\frac{z_i}{\sqrt{d_{ii}}} - \frac{z_j}{\sqrt{d_{jj}}} \right)^2. \end{aligned} \tag{2.7}$$

Esta última expresión muestra que $\mathbf{z}^T L \mathbf{z}$ mide la diferencia al cuadrado entre las predicciones z_i y z_j , ponderados por a_{ij} . Si las secuencias i y j son similares, y por lo tanto a_{ij} tiene un valor relativamente alto, cualquier diferencia entre las dos predicciones tendrá un alto costo. Si no hay arista que conecte las dos secuencias, $a_{ij} = 0$ la diferencia entre las predicciones se ignora. Además, se debe tener en cuenta que las predicciones se ponderan por el inverso de la raíz cuadrada del grado del nodo. Como resultado, los nodos con un grado pequeño se consideran tan importantes como los nodos altamente conectados.

Entonces, la función objetivo tiene dos componentes: el primer término mide la falta de suavidad de la solución usando la matriz laplaciana normalizada (componente no supervisada del aprendizaje), y el segundo término es la diferencia al cuadrado entre predicciones y etiquetas distintas de cero en ℓ (componente supervisada). Para aprovechar al máximo el aprendizaje semi-supervisado, no debe haber una gran superposición entre las clases que se separarán. Sin embargo, si este requisito previo no se cumple, el primer término de la función objetivo no tendrá ningún mínimo importante. Por lo tanto, el segundo término de la ecuación (el supervisado) conducirá la búsqueda. De esta forma, si no hay una separación clara entre las clases, el método se comportará de forma similar a cualquier otro método supervisado en las mismas condiciones.

El problema de optimización completo queda definido como

$$\begin{aligned} \arg \min_{\mathbf{z}} \quad & \mathbf{z}^T L \mathbf{z} + c(\mathbf{z} - \ell)^T C(\mathbf{z} - \ell) \\ \text{s.t.} \quad & \mathbf{z}^T \mathbf{1} = 0, \\ & \mathbf{z}^T \mathbf{z} = n \end{aligned} \quad (2.8)$$

donde la combinación de ambas restricciones evita soluciones triviales. En la primera restricción se requiere que la suma de los elementos de \mathbf{z} sea cero; es decir, las etiquetas de predicción deben tener tanto valores negativos como positivos. La segunda restricción elimina las versiones escaladas de la solución que, para nuestro propósito, son todas equivalentes.

Los valores ℓ_i se establecen en γ_+ , γ_- o cero, dependiendo de si la secuencia i -th es positiva, negativa o desconocida, respectivamente. Como la función objetivo obliga a los valores de \mathbf{z} a acercarse a γ_+ o γ_- , estas constantes deben definirse de manera tal que la solución óptima \mathbf{z}^* pueda satisfacer ambas restricciones del problema. Si n_+ y n_- son las cantidades reales de nodos positivos y negativos en la solución, definiendo $\gamma_+ = \sqrt{n_-/n_+}$ y $\gamma_- = -\sqrt{n_+/n_-}$ se consigue que \mathbf{z}^* satisface ambas restricciones. Esto se puede observar al reemplazar \mathbf{z}^* en las restricciones y asumiendo que este vector tiene n_+ elementos iguales a γ_+ y n_- iguales a γ_- . Los números n_+ y n_- son generalmente desconocidos, pero se pueden estimar fácilmente a partir de los ejemplos disponibles. Si se proporcionan ejemplos positivos y negativos para el entrenamiento, se calculará n_+/n_- como la proporción de ejemplos dados. Si sólo hay ejemplos positivos, n_+ se estima como el doble del número de secuencias de entrenamiento y $n_- = n - n_+$. Esta estimación podría mejorarse utilizando el conocimiento del dominio, es decir, utilizando el número esperado de miARNs para una especie determinada; sin embargo, no es necesario ya que el método propuesto no es sensible a estos parámetros (ver Figura S1 en el Material Suplementario del Anexo 5). Por lo tanto, cualquier valor entre el número de secuencias de entrenamiento positivas y cuatro veces este número se puede utilizar sin impacto en el rendimiento.

La constante c en la función objetivo se puede usar para establecer el peso relativo del segundo término en comparación con el primero. Un valor grande de c otorga una mayor penalización a las clasificaciones erróneas, lo que lleva los puntajes de predicción a valores similares a las etiquetas distintas de cero ℓ . Por el contrario, si se usa un valor bajo de c , las clasificaciones erróneas se penalizan menos y el primer término domina la función objetivo, lo que produce una solución más suave. La matriz C en el segundo término es una matriz diagonal que tiene valor cero en los elementos correspondientes a secuencias desconocidas. De esta manera, las secuencias sin etiqueta se ignoran en este término. En los elementos distintos de cero (correspondientes a los ejemplos etiquetados), el valor asignado permite diferentes penalizaciones por clasificación errónea para cada secuencia. Esta ponderación puede usarse, por ejemplo, para asignar valores inferiores a los pre-miARN que no han sido validados experimentalmente o que son ejemplos negativos poco confiables. También se puede usar para evitar la clasificación errónea de ejemplos etiquetados. En la Sección S2 del Material Suplementario del Anexo 5 se demuestra que si $C_{ii} > (nn_+)/(cn_-)$, la clasificación incorrecta de la secuencia i -ésima tendrá una mayor penalización que cualquier penalización en el término no supervisado. Entonces, no puede ser mal clasificado. Como valor predeterminado los elementos distintos de cero de C se establecen en 1, tanto para los ejemplos positivos como negativos.

Para resolver este problema de optimización primero calculamos la descomposición espectral del Laplaciano $L = U\Sigma U^T$. A continuación, utilizamos un nuevo vector de parámetro \mathbf{w} tal que $\mathbf{z} = U\mathbf{w}$. Como el vector propio correspondiente al valor propio más bajo siempre es constante, la primera

restricción del problema de optimización se convierte en $w_1 = 0$. Si definimos V como la matriz con todos los vectores propios, excepto el primero, y H como la matriz diagonal con todos los valores propios, excepto el más bajo, obtenemos el siguiente problema de optimización

$$\begin{aligned} \arg \min_{\mathbf{w}} \quad & \mathbf{w}^T H \mathbf{w} + c(V \mathbf{w} - \boldsymbol{\ell})^T C(V \mathbf{w} - \boldsymbol{\ell}) \\ \text{s.t.} \quad & \mathbf{w}^T \mathbf{w} = n. \end{aligned} \quad (2.9)$$

Definiendo $Q = H + cV^T C V$ y $b = cV^T C \boldsymbol{\ell}$, este problema puede reescribirse como

$$\begin{aligned} \arg \min_{\mathbf{w}} \quad & \mathbf{w}^T Q \mathbf{w} + 2b^T \mathbf{w} + c\boldsymbol{\ell}^T C \boldsymbol{\ell} \\ \text{s.t.} \quad & \mathbf{w}^T \mathbf{w} = n, \end{aligned} \quad (2.10)$$

donde el último término se puede descartar dado que es constante. Usando los multiplicadores de Lagrange el mínimo global de esta función se produce en $\mathbf{w}^* = (Q - \lambda^* I)^{-1} \mathbf{b}$. Ahora, usando los resultados de Gander *et al.* (1989), λ^* se puede calcular como el menor autovalor de

$$M = \begin{pmatrix} Q & -I \\ \frac{\mathbf{b}\mathbf{b}^T}{n} & Q \end{pmatrix}. \quad (2.11)$$

A partir de esta solución, las etiquetas se calculan como $\mathbf{z}^* = V \mathbf{w}^*$.

2.3.4. Umbralización de los puntajes de predicción

Dados los altos desbalances que están presentes en los datos de un genoma completo, es necesario aumentar el costo de clasificación errónea de la clase positiva. Mientras que la matriz C se puede usar para asignar diferentes costos de clasificación errónea en el proceso de optimización, estimar el umbral que se aplicará en \mathbf{z} es un método más flexible ya que permite optimizar una única vez y luego elegir la medida de rendimiento a maximizar. Dado que se espera que los puntajes de predicción ordenen las secuencias de acuerdo a la probabilidad de ser pre-miARNs, modificar el umbral utilizado para separar las dos clases equivale a clasificar con distintas ponderaciones por clase (Mease *et al.*, 2007).

Una métrica de evaluación común que se usa en problemas con desbalance de clases es la media geométrica (\bar{G}) de la sensibilidad y la especificidad $\bar{G} = \sqrt{S^+ S^-}$ (Batuwita and Palade, 2009; Gudys *et al.*, 2013), donde S^+ es la proporción de secuencias positivas correctamente clasificadas como positivas (sensibilidad), y S^- es la proporción de secuencias negativas correctamente clasificadas como negativas (especificidad). Esta medida tiene la ventaja de dar la misma importancia a las clases negativas y positivas, independientemente de la cantidad de elementos en cada clase. Una mejor medida es el F-score, $F_1 = 2PS^+/(P + S^+)$, donde P es la precisión (la proporción de verdaderos positivos dentro de las secuencias clasificadas como positivas). Cuando se utiliza F_1 para la optimización del umbral se obtiene una mejor P a costa de una S^+ más baja. Debido al gran desbalance de clases en este problema de clasificación, es importante prestar especial atención a la cantidad de falsos positivos. Por lo tanto, F_1 es una mejor medida para este problema.

Los puntajes de predicción obtenidos para los ejemplos etiquetados se pueden usar para encontrar el umbral que maximice cierta medida de rendimiento objetivo. Debe señalarse que esto es sólo una estimación, ya que se desconoce la clase real a la que pertenecen las secuencias no etiquetadas. Por esta razón, si ordenamos los puntajes de predicción \mathbf{z}^* y calculamos la medida de desempeño utilizando como umbral cada valor de este vector, entre dos ejemplos etiquetados consecutivos aparecerán regiones donde esta medida se hace constante. Si el número de ejemplos etiquetados es bajo, estas regiones constantes pueden ser relativamente grandes. Por lo tanto, el umbral final se establece como el punto medio entre el puntaje de predicción más alto y más bajo donde se maximice la medida de rendimiento (ver Figura S3 del anexo 5).

Sección 3

Resultados

3.1. Procesamiento de secuencias de ARN de tipo tallo-horquilla

Se ha desarrollado una herramienta llamada HextractoR, simple e integrada, que automáticamente extrae y pliega todas las secuencias de horquilla a partir de datos brutos del genoma completo. El método propuesto aprovecha los últimos avances en la predicción de estructuras secundarias. Al procesar grandes ventanas superpuestas, no se pierden ni se cortan de manera inapropiada las horquillas. Además, permite procesar genomas en paralelo y con pocos requisitos de memoria, ya que puede dividir automáticamente archivos de entrada demasiado grandes.

Se procesaron varios genomas y los resultados se compararon con los del programa mirCheck de EMBOSS (Olson, 2002). La Tabla 3.1 muestra los resultados del procesamiento de 5 genomas completos: *Homo sapiens*, *Arabidopsis thaliana*, *Danio rerio*, *Anopheles gambiae* y *Caenorhabditis elegans*, además de otra especie menos estudiada *Echinococcus multilocularis*. Para cada uno se muestra el número de secuencias extraídas con mirCheck y HextractoR, en la segunda y tercera columna, respectivamente. Con respecto a los pre-miARNs en particular se informa, en cada genoma analizado, en la cuarta columna de la tabla el número de pre-miARNs conocidos (de acuerdo con miRBase 21) para las 6 especies. La proporción de aquellos encontrados por los métodos comparados se muestra en las dos últimas columnas. Se puede ver claramente que en todos los casos el rendimiento de HextractoR es superior. Además, en muchos casos, el 100 % de los pre-miARNs conocidos se extraen correctamente.

Las diferencias en cantidad de pre-miARNs encontrados que muestra la Tabla 3.1 se pueden deber a varios motivos. En primer lugar, como se ve en la segunda y tercer columna, HextractoR consigue capturar más horquillas que mirCheck. Al utilizar ventanas solapadas de un tamaño varias veces mayor al de una horquilla para recorrer el genoma, HextractoR consigue capturar cualquier horquilla. MirCheck se podría configurar para que utilice ventanas con una mayor longitud, pero en ese caso descartaría la mayoría de las secuencias porque estas formarían estructuras con varios bucles y mirCheck no realiza el proceso de separación en horquillas como si lo hace HextractoR.

3.2. Predicción de pre-microARN

En esta sección se presentan los resultados de los experimentos realizados para probar el rendimiento de miRNAss en diferentes condiciones. En la primera subsección, se reprodujeron experimentos realizados por otros autores para comparar miRNAss con métodos supervisados del estado del arte en condiciones controladas. Los conjuntos negativos fueron definidos artificialmente por los autores originales y en este caso la proporción de ejemplos etiquetados fue muy alta en relación a la que se puede encontrar en genomas reales. En la segunda subsección el porcentaje de ejemplos etiquetados se redujo gradualmente para acercarse más a un caso real de predicción de pre-miARN. Además, miRNAss se probó en un esquema de una clase, donde sólo se conocían ejemplos positivos (pre-miARNs reales) de antemano. En la última subsección, miRNAss se aplicó a una tarea de predicción real a partir de datos del genoma completo de 3 especies modelo, sin ejemplos negativos y utilizando sólo un bajo número

Especies	Horquillas extraídas		pre-miARNs encontrados		pre-miARNs
	Einverted	HextractoR	Einverted	HextractoR	conocidos
<i>Anopheles gambiae</i>	1.410.532	4.276.543	92,42 %	100,00 %	66
<i>Caenorhabditis elegans</i>	875.588	1.739.124	90,00 %	99,60 %	250
<i>Danio rerio</i>	11.028.128	23.214.338	93,06 %	99,42 %	346
<i>Echinococcus multilocularis</i>	509.530	1.898.911	81,81 %	100,00 %	22
<i>Arabidopsis thaliana</i>	874.320	1.357.455	91,69 %	94,77 %	325
<i>Homo sapiens</i>	18.654.426	48.206.494	85,38 %	97,45 %	1881

Tabla 3.1: Cantidad de horquillas y pre-miARNs encontrados por cada método en el genoma de 6 especies

de ejemplos positivos. En esta última prueba había un enorme desbalance de clases y se procesaron más de un millón de secuencias de entrada.

3.2.1. Comparación con métodos del estado del arte

Se estableció F_1 como medida de desempeño utilizada para la optimización del umbral en todos los métodos. Los experimentos se diseñaron para estimar el desempeño de métodos de aprendizaje supervisado, por lo que se utiliza un esquema estándar de validación cruzada con 10 particiones. En este esquema, la mayoría de las secuencias están etiquetadas artificialmente para el entrenamiento, y sólo un 10 % queda sin etiquetar. Por lo tanto, es importante señalar que la principal ventaja de miRNAss, que es explotar datos no etiquetados para mejorar los resultados, no se puede aprovechar en estos experimentos. Para una comparación justa, en estos experimentos los clasificadores usan las mismas características. Este conjunto está compuesto por 7 características de miPred (Ng and Mishra, 2007), más 14 de microPred (Batuwita and Palade, 2009) y 7 añadidas por Gudyś *et al.* (2013). Para obtener más detalles sobre el conjunto de características, se puede consultar la Sección S4 del Anexo 5.

Como antes se especificó, se ejecutó una validación cruzada estratificada de 10 particiones en cinco conjuntos de datos y se compararon los resultados con los obtenidos por HuntMi (Gudyś *et al.*, 2013). Las particiones fueron construidas al azar usando una semilla fija para tener experimentos totalmente reproducibles. Tres conjuntos de datos contenían una mezcla de secuencias de diferentes especies, que se agruparon en los conjuntos de datos animales (10 especies), plantas (7 especies) y virus (29 especies). Los otros dos conjuntos de datos contenían secuencias de *Homo sapiens* y *Arabidopsis thaliana*. Los parámetros utilizados en miRNAss fueron los mismos en todas las pruebas: $c = 1$ y $k = 10$ (tanto para RELIEF-F como para la construcción de grafos). MiRNAss se probó con y sin RELIEF-F, para medir su impacto en el rendimiento. La ponderación de características se calculó utilizando sólo los datos de entrenamiento de cada partición.

Los tiempos de computo medidos ¹ en cada partición se promediaron y se presentan en la Tabla 3.2. La primera columna muestra el número total de secuencias. La segunda y tercera columnas indican el número de secuencias pre-miARN y no pre-miARN en cada conjunto de datos. La cuarta y quinta columnas presentan los tiempos que demoró cada método. La última columna muestra el cociente entre el tiempo que tardó HuntMi y el tiempo que tardó miRNAss. La mayor diferencia se observó en el conjunto de datos más pequeño (virus): miRNAss fue 38 veces más rápido que HuntMi. Arabidopsis es el segundo conjunto de datos más pequeño, y aquí miRNAss fue 14 veces más rápido. En los siguientes tres conjuntos de datos las diferencias de tiempo de computación entre miRNAss y HuntMi aumentaron a medida que crecía el número de secuencias. Estos resultados muestran que miRNAss no sólo es mucho más rápido sino que el costo computacional crece más rápido en HuntMi. Tales diferencias pueden ser irrelevantes en pequeños conjuntos de datos como los que se utilizan en esta subsección, pero se vuelven muy importantes en tareas de predicción de genoma completo, donde se procesan varios millones de secuencias. Por ejemplo, si extrapolamos los tiempos de entrenamientos utilizando los datos de la tabla, para entrenar a HuntMi con 1.7 millones de secuencias (uno de los conjuntos de datos del

¹Intel ®Core™ i5-4460 CPU @ 3.20GHz, 8 GB de RAM.

Conjunto de datos	Número de secuencias			Tiempos		
	Total	miARN	no-miARN	HuntMi	miRNAss	speedup
Virus	1.076	237	839	38 s	1 s	38,00
Arabidopsis	28.590	231	28.359	14819 s	129 s	14,10
Human	82.634	1.406	81.228	94873 s	462 s	21,37
Plants	117.101	2.172	114.929	214561 s	714 s	30,19
Animals	225.207	7.053	218.154	654762 s	1.834 s	35,85

Tabla 3.2: Comparación de los tiempos de ejecución entre miRNAss y HuntMi.

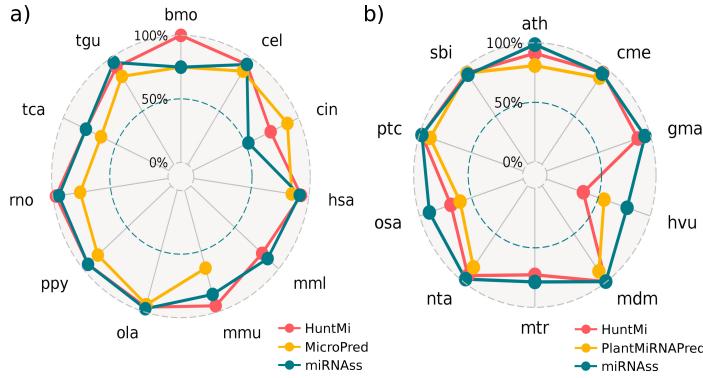


Figura 3.1: Sensibilidad obtenida con varios clasificadores del estado del arte en: a) animales, y b) plantas. La distancia de cada punto al centro mide la sensibilidad obtenida cada especie.

genoma utilizado en la sección 3.2.3) se necesitarían alrededor de 37 días, mientras que miRNAss sólo se requieren 18 horas.

Con respecto a las medidas de predicción, ambos métodos se desempeñaron de manera similar en este experimento. Se aplicó una prueba de Friedman (1937), que dio como resultado un valor p de 0,179. Esto demuestra que el método propuesto puede obtener resultados equivalentes a un método del estado del arte para la configuración supervisada, aunque en menos tiempo (como se muestra en la Tabla 3.2). Además, fue posible verificar que RELIEF-F mejoró los resultados en todas las pruebas, aunque en algunas de ellas las diferencias fueron pequeñas. Esto era de esperar, dado que las características utilizadas en estos conjuntos de datos son el resultado de procesos previos de selección de características. Una prueba de Friedman en esta comparación da como resultado un valor p de 0.025, lo que demuestra que las diferencias son significativas. Se pueden ver más detalles sobre estos resultados en la Figura S5 del Anexo 5.

Para probar la capacidad de miRNAss para predecir nuevos pre-miARNs en diferentes especies, los pre-miARNs conocidos de animales y plantas que se incluyeron en la versión 17 de mirBase se usaron para el entrenamiento, y los pre-miARNs que se agregaron en las versiones 18 a 19 fueron usados como conjunto de prueba. En el caso de las especies animales, se añadió microPred (Batuwita and Palade, 2009) en la comparación, ya que era el mejor software para la predicción de miARN humano en el momento de su publicación. También se agregó el método PlantMiRNAPred (Xuan *et al.*, 2011b), porque está específicamente diseñado para especies de plantas. Cabe señalar que, una vez más, el porcentaje de secuencias no etiquetadas en el conjunto de datos es muy bajo: menos del 0,3 % en animales y menos del 1,3 % en plantas. La S^+ obtenida para cada especie se muestran en la Figura 3.1, donde cada clasificador se muestra con un color diferente a través de una gráfica de radar. En el conjunto de datos de animales, miRNAss superó a microPred en casi todas las especies. En el conjunto de datos de humanos, miRNAss obtuvo una mayor S^+ que microPred, el cual, como se mencionó anteriormente, se diseñó específicamente para humanos. Comparado con HuntMi, miRNAss obtuvo mayores S^+ en 4 especies, HuntMi produjo mejores resultados en 4 especies y los resultados obtenidos fueron iguales en 3 especies. En plantas, miRNAss superó a PlantMiRNAPred en casi todas

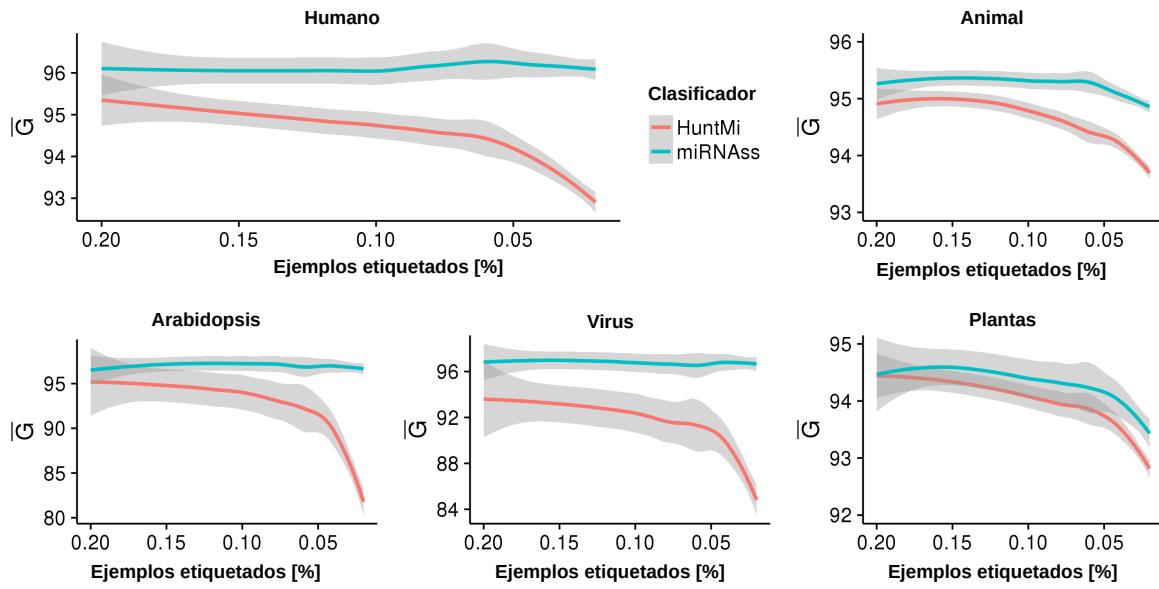


Figura 3.2: Curvas de \bar{G} obtenidas al disminuir el porcentaje de ejemplos etiquetados. Regiones sombreadas representan los intervalos de confianza de la estimación con regresión local (LOESS) con $p < 0,05$.

las especies. Comparado con HuntMi, miRNAss tuvo una mejor S^+ en 8 especies, mientras que HuntMi superó ligeramente al miRNAss en sólo 2 especies (*Cucumis melo* y *Sorghum bicolor*). Para analizar si hubo diferencias significativas entre los clasificadores, se realizaron pruebas de Nemenyi (1962) para ambos conjuntos de datos ($p < 0,05$). En las especies de animales, HuntMi y miRNAss produjeron resultados equivalentes, pero ambos se desempeñaron mejor que microPred. En especies de plantas, miRNAss obtuvo el rango más bajo (el mejor) y una diferencia significativa con PlantMiRNAPred. Sin embargo, la diferencia con HuntMi no fue estadísticamente significativa.

Los métodos supervisados hacen un uso extenso de los datos etiquetados, no sólo para el entrenamiento sino también para encontrar los hiper-parámetros y umbrales óptimos. Por el contrario, miRNAss se diseñó sobre la hipótesis de que los ejemplos etiquetados son escasos, poco confiables y no representativos de toda la clase, que en realidad es un escenario más realista para esta tarea de predicción. Por lo tanto, debe tenerse en cuenta que, incluso bajo estas condiciones desfavorables, miRNAss obtuvo resultados significativamente mejores que MicroPred y PlantMiRNAPred, y resultados equivalentes a los producidos por HuntMi, siendo sin embargo, mucho más rápido.

3.2.2. Pocos ejemplos etiquetados

Para representar un escenario más realista en el que el número de ejemplos conocidos es muy bajo, los cinco conjuntos de datos utilizados en las últimas pruebas se utilizaron con distintos porcentajes de secuencias etiquetadas y se realizaron pruebas en un esquema de validación entrenamiento-prueba. El porcentaje de ejemplos etiquetados se redujo del 20 % al 2 %, con un paso del 2 %. Los ejemplos etiquetados se seleccionaron al azar y las pruebas se repitieron 200 veces para cada porcentaje para estimar los intervalos de confianza. En la Figura 3.2, se estimaron las curvas de F_1 esperadas con intervalos de confianza de 0,05 para la comparación. En el conjunto de datos humanos, el F_1 es casi un 10 % más alto para miRNAss, independientemente del porcentaje de secuencias marcadas. En el conjunto de datos de *Arabidopsis*, donde el número de secuencias de ejemplos positivos fue menor, las diferencias a favor de miRNAss fueron más altas en los porcentajes más bajos de ejemplos etiquetados, lo que indica que miRNAss puede identificar efectivamente la clase positiva incluso con un número muy bajo de ejemplos. Las mismas tendencias se observan en los conjuntos de datos de animales, plantas y virus. Estos resultados no sólo muestran que miRNAss es capaz de superar a los métodos

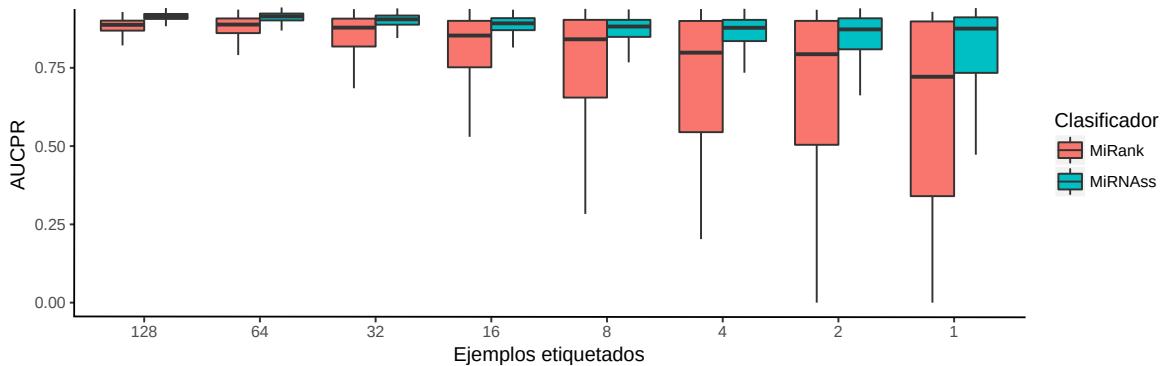


Figura 3.3: Diagrama de caja de las AUC obtenidas por miRNAss y miRank, con diferentes cantidades de ejemplos de entrenamiento positivos.

supervisados cuando el número de ejemplos etiquetados es bajo, sino también que las tasas de error estimadas usando una alta proporción de ejemplos etiquetados son muy diferentes de las obtenidas en escenarios más realistas.

Un paso más hacia una tarea de predicción más realista consiste en usar sólo ejemplos positivos. Bajo estas condiciones, miRNAss se comparó con miRank (Xu *et al.*, 2008), que fue diseñado para trabajar con un número extremadamente pequeño de ejemplos positivos. Se utilizaron los conjuntos de datos proporcionados por el autor original: 533 pre-miARNs humanos y 1000 secuencias no miARN. Para hacer una comparación justa, ambos métodos usaron el conjunto de características de miRank. Este conjunto está compuesto por 32 tripletas (Xue *et al.*, 2005), MFE normalizado, propensiones de emparejamiento de bases normalizadas de ambos brazos y longitud de bucle normalizado. Se marcó un número variable de ejemplos positivos (1, 2, 4, 8, 16, 32, 64 y 128) y el resto de las secuencias se dejaron sin etiquetar para medir las tasas de error. Como los resultados dependen de un muestreo aleatorio de las secuencias, este procedimiento se repitió 1000 veces para cada número de ejemplos etiquetados. Dado que MiRank utiliza como resultado una puntuación continua, para comparar también se utilizaron los puntajes de predicción obtenidos con miRNAss en lugar de utilizar las clases. Se calculó el área bajo la curva de precisión-recuperación (AUCPR, del inglés *Area Under Precision-Recall Curve*), para realizar una comparación independiente del umbral utilizado para definir las clases (Bradley, 1997). La Figura 3.3 presenta un diagrama de caja con la distribución de AUCPR obtenida por cada clasificador para diferentes cantidades de ejemplos etiquetados. MiRNAss mantuvo una AUCPR casi constante, independientemente del número de pre-miARNs marcados, mientras que el rendimiento de miRank disminuyó marcadamente. Además, los valores de AUCPR para miRNAss mostraron una pequeña dispersión en comparación con miRank, que se vuelve muy inestable cuando disminuye el número de ejemplos etiquetados. Esta inestabilidad puede ser producida por ejemplos positivos que están cerca de la frontera de la clase, lo que hace que miRank no establezca correctamente la frontera de decisión. El algoritmo semi-supervisado de miRNAss logró encontrar correctamente las regiones de baja densidad que separan los miARNs del resto de las secuencias, independientemente de los ejemplos proporcionados.

3.2.3. Predicción real en genomas completos

MiRNAss se probó con los genomas completos de tres especies bien conocidos: *Arabidopsis thaliana*, *Caenorhabditis elegans* y *Anopheles gambiae*, para reproducir todas las condiciones de una tarea de predicción real. Los genomas completos se procesaron para extraer todos las secuencias con estructura secundaria tipo tallo-horquilla existentes. Para este propósito se utilizó HextractoR, la herramienta desarrollada para tal fin. Este proceso dejó un total de 1.356.616 secuencias de *A. thaliana*; 1.698.859 secuencias de *C. elegans*; y 4.276.188 secuencias de *A. gambiae*. Para detectar los pre-miARNs conocidos se utilizó la base de datos mirBase v21. Esto definió un total de 304, 249 y 66 horquillas

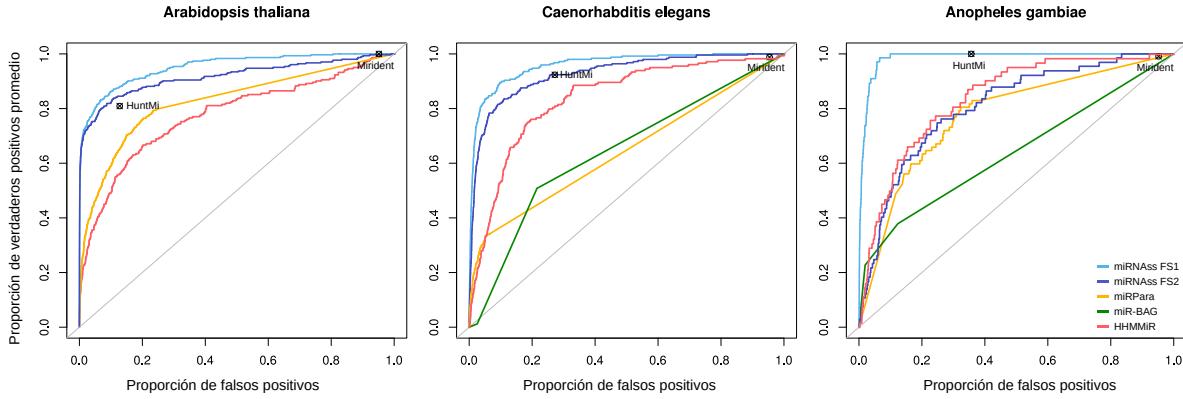


Figura 3.4: Curvas ROC de miRNAss y otros métodos del estado del arte en datos de genoma completo de tres especies. Los puntos muestran el desempeño alcanzado por los métodos que generan clasificaciones sin puntajes de predicción.

como conjunto positivo para *A. thaliana*, *C. elegans* y *A. gambiae*, respectivamente. Se extrajeron dos conjuntos de características de los tres genomas. El primero (FS1) es el mismo usado en la Sección 3.1, para hacer una comparación justa con uno de los métodos. El segundo conjunto de características (FS2) es un conjunto ampliado compuesto por casi todas las características propuestas para la predicción pre-miARN en la literatura. FS2 se calculó con miRNAfe y cada vector de característica resultó en 79 elementos. Para obtener más información sobre los conjuntos de características, se puede consultar la Sección S4 del Anexo 5. Se utilizó un esquema de validación cruzada de 10 particiones para medir el rendimiento con curvas ROC promediadas. Para comparar el rendimiento con miRNAss se probaron muchos algoritmos de predicción de pre-miARN en estos conjuntos de datos. En total se han probado once predictores, pero la mayoría han fallado con datos de genomas completos o sus servidores no funcionan. Los cinco predictores que se pudieron aplicar a genomas completos fueron HuntMi, Mirident (Liu *et al.*, 2012), HHMMiR (Kadri *et al.*, 2009), miRPara (Wu *et al.*, 2011) y miR-BAG (Jha *et al.*, 2012). Los primeros dos métodos producen clasificaciones duras (en vez de puntajes continuos) por lo que se representan como puntos en las figuras ROC. Por otro lado, dado que miRPara, miR-BAG, HHMMiR y miARN proporcionan puntajes continuos, para estos casos se pudieron obtener las curvas ROC completas. MiR-BAG no se ejecutó en *A. thaliana* porque no proporciona un modelo pre-entrenado para plantas. Los puntajes de salida obtenidos con este método sólo pueden tomar cuatro valores posibles, por lo que las curvas ROC contienen regiones constantes. Estos resultados se presentan en la Figura 3.4.

En el genoma de *A. thaliana*, se puede ver que las curvas ROC de miRNAss están por encima de las curvas de miRPara y HHMMiR para todos los valores umbral. Cabe señalar que Mirident, a pesar de tener la mayor tasa de positivos verdaderos (sensibilidad), también tiene la mayor tasa de falsos positivos. HuntMi es más equilibrado, con un alto reconocimiento de secuencias positivas y un número moderado de falsos positivos. Sin embargo, está debajo de miRNAss con cualquiera de los dos conjuntos de características. En el genoma de *C. elegans* se puede hacer un análisis similar para HuntMi y Mirident. En este conjunto de datos, miR-BAG genera una curva ROC similar a la curva de MiRPara, ambas debajo del resto de las curvas. HHMMiR presenta un mejor rendimiento que estos métodos, pero una vez más es superado por miRNAss. En el caso del genoma de *A. gambiae*, el rendimiento de miRNAss con FS1 es más distante al obtenido con FS2. MiR-BAG y HHMMiR generan una curva similar a la obtenida por miRNAss con FS2, muy por debajo de la obtenida con FS1. La curva ROC con FS1 muestra, en la esquina superior izquierda, que miRNAss puede proporcionar el mejor equilibrio entre sensibilidad y tasa de falsos positivos. De hecho, esta es casi una curva ROC ideal.

Finalmente, como resumen del análisis comparativo, la Tabla 3.3 presenta más resultados de interés práctico. Los mismos métodos y especies de la Figura 3.4 se analizan aquí según el rendimiento global y el número total de candidatos que devuelve cada método, es decir, la suma de verdaderos positivos y

Tabla 3.3: Media geométrica de la sensibilidad y la especificidad (\bar{G}) y suma de falsos y verdaderos positivos ($TPFP$, por sus siglas en inglés) en las tres pruebas de genoma completo.

Classifier	<i>A. thaliana</i>		<i>C. elegans</i>		<i>A. gambiae</i>	
	$TPFP$	\bar{G}	$TPFP$	\bar{G}	$TPFP$	\bar{G}
Mirident	1.294.648	22,05 %	1.617.221	21,29 %	4.068.431	21,86 %
miR-BAG	-	-	375.011	63,14 %	495.231	57,62 %
miRPara	2.755	47,95 %	11.712	53,79 %	283.232	72,48 %
HHMMiR	45.104	69,07 %	40.318	73,29 %	91.093	74,07 %
HuntMi	173.906	84,00 %	462.203	82,00 %	1.456.590	80,20 %
MiRNAss	134.369	84,82 %	164.557	87,61 %	258.096	93,34 %

falsos positivos ($TPFP$). Se puede ver que miARNs supera a todos los métodos en los tres genomas. Mirident es el método con el rendimiento más bajo para todas las especies. Esto se debe a que este método etiqueta como positivos a casi todos los ejemplos, lo que se refleja en una sensibilidad muy alta, pero sin utilidad práctica dada la cantidad de candidatos proporcionados. MiR-BAG tiene un mejor pero aún pobre desempeño en ambas especies. HHMMiR y miRPara predicen muy pocos candidatos, con una alta especificidad a costa de una sensibilidad muy baja. HuntMi, en cambio, permite obtener resultados más equilibrados, con el segundo mejor rendimiento. Sin embargo, en *A. gambiae* devuelve una cantidad de falsos positivos más de 5 veces mayor que los devueltos por MiRNAss.

Estos resultados nos permiten afirmar que MiRNAss supera a los métodos del estado del arte en una configuración de clasificación realista. Los ejemplos negativos definidos artificialmente se usan para entrenar modelos supervisados y, dado que estos ejemplos no son representativos de la gran diversidad de la clase negativa, los modelos no descartan correctamente secuencias que no sean miARN. Por el contrario, MiRNAss puede aprovechar mejor el gran número de secuencias no etiquetadas para ajustar mejor la frontera de decisión alrededor de los pre-miARN, descartando el resto de las secuencias.

Sección 4

Conclusiones

En esta tesis doctoral, presentamos una nueva metodología de predicción pre-miARN compuesta por tres etapas. Para la primer etapa se desarrolló un método para extraer secuencias con estructura secundaria tipo horquilla del genoma completo. Este método contempla además el recorte de las secuencias y la eliminación de horquillas repetidas, tanto idénticas como similares. Las pruebas en los genomas de 6 especies demostraron que el método propuesto encuentra una mayor cantidad de horquillas que el único método disponible para esta tarea y, lo que es más importante aún, una mayor cantidad de pre-miARNs. El método con el cual se comparó encontró en cada genoma aproximadamente un 90 % de los pre-miARNs conocidos llegando a un 80 % en el caso de una especie no modelo. En cambio la metodología propuesta alcanzó valores cercanos al 100 % en todas las especies.

En segundo lugar, se realizó una exhaustiva revisión del estado del arte para analizar que características se utilizan en la predicción de miARNs. Con esta información se desarrolló una biblioteca de algoritmos de extracción de características y una versión web para utilizar estos algoritmos sin la necesidad de tener conocimientos sobre programación. Consideramos además que la revisión del estado del arte y la posterior recopilación de información puede ser de gran valor para la comunidad.

Por último, se desarrolló un algoritmo de predicción que utiliza un enfoque semi-supervisado para enfrentar el problema de ejemplos de entrenamiento escasos y poco confiables. Los experimentos realizados en una configuración supervisada forzada mostraron que alcanza las tasas de clasificación de los mejores métodos del estado del arte en pruebas de validación cruzada estándar y en tiempos más cortos. El método propuesto también se probó en condiciones que están más cerca de una tarea de predicción real, donde se reduce el número de secuencias etiquetadas. En estas pruebas, superó claramente al mejor método supervisado disponible, el cual sufrió una importante caída de desempeño al reducirse la cantidad de ejemplos positivos disponibles. Además se obtuvieron mejores resultados que otro método diseñado especialmente para trabajar con pocos ejemplos de entrenamiento. Para la última prueba se procesaron tres genomas completos de especies modelo y se compararon resultados con varios métodos del estado del arte. En esta prueba una gran cantidad de métodos fallaron porque no fueron capaces de procesar la gran cantidad de secuencias de un genoma completo. Los métodos que funcionaron obtuvieron desempeños por debajo del obtenido con el método desarrollado, en algunos casos cercanos al error de un clasificador al azar.

Los resultados a lo largo de este trabajo permiten arribar a las siguientes conclusiones:

- Existía una necesidad de mejorar el proceso de extracción de horquillas del genoma completo, dado que los métodos existentes no se desempeñaban satisfactoriamente y una gran cantidad de pre-miARNs se perdían en este temprano paso. La metodología diseñada logró superar esta falencia.
- La biblioteca de extracción de características desarrollada permite calcular la gran mayoría de las características utilizadas en la predicción de miARNs en la actualidad.
- El método de predicción desarrollado demostró ser escalable, a diferencia de los métodos del estado del arte que en su mayoría no lograron funcionar con genomas completos.

- Los ejemplos negativos que se utilizan para entrenar muchos métodos de predicción del estado del arte no son representativos de la clase completa de secuencias no miARN.
- Los métodos supervisados de predicción de miARN logran tasas de desempeño muy altas en pruebas de validación cruzada con una clase negativa definida artificialmente, pero el rendimiento disminuye en gran medida cuando tienen que enfrentar la diversidad de secuencias que se pueden encontrar en un genoma real. Por lo tanto, esta metodología de validación no permite obtener conclusiones confiables sobre el desempeño de las distintas técnicas de predicción.
- El método de predicción desarrollado resulta eficiente y escalable, lo que permite procesar genomas completos con varios millones de horquillas.
- El método de predicción desarrollado busca automáticamente una amplia variedad de ejemplos negativos entre las secuencias tipo horquilla. Ademas, al ser un algoritmo de aprendizaje semi-supervisado, tiene en cuenta la distribución de las secuencias sin etiqueta en el espacio de las características para ajustar fronteras de decisión alrededor de los pre-miARNs. Esto permite un mejor desempeño que el de los métodos del estado del arte.

Sección 5

Publicaciones

A continuación se listan todos los trabajos relacionados con la predicción de microARN en los que se participó durante el desarrollo del doctorado.

Publicaciones en revistas

- **Yones, C. A.**, Stegmayer, G., Kamenetzky, L., & Milone, D. H. (2015). miRNAfe: a comprehensive tool for feature extraction in microRNA prediction. *Biosystems*, 138, 1-5.
- Kamenetzky, L., Stegmayer, G., Maldonado, L., Macchiaroli, N., **Yones, C.**, & Milone, D. H. (2016). MicroRNA discovery in the human parasite *Echinococcus multilocularis* from genome-wide data. *Genomics*, 107(6), 274-280.
- Stegmayer, G., **Yones, C.**, Kamenetzky, L., & Milone, D. H. (2017). High class-imbalance in pre-miRNA prediction: a novel approach based on deepSOM. *IEEE/ACM transactions on computational biology and bioinformatics*, 14(6), 1316-1326.
- **Yones, C.**, Stegmayer, G., & Milone, D. H. (2017). Genome-wide pre-miRNA discovery from few labeled examples. *Bioinformatics*. 34(4), 541-549.
- Bugnon, L., **Yones, C.**, Milone, D. H., y Stegmayer, G. (2017). Novel som architectures for large class imbalance prediction in genome-wide data. Trabajo enviado a IEEE/ACM Transactions on Computational Biology and Bioinformatics.
- Stegmayer, G., Di Persia, L., Rubiolo, M., Gerard, M., Pividori, M., **Yones, C.**, Bugnon, L., Rodriguez, T., Raad, J., y Milone, D. (2018). Predicting novel microrna: a comprehensive comparison of machine learning approaches. *Briefings in Bioinformatics*.
- Bugnon, L., **Yones, C.**, Milone, D. H., y Stegmayer, G. (2018). Deep neural architectures for highly imbalanced data in bioinformatics. Trabajo enviado a IEEE Transactions on Neural Networks and Learning Systems, Special Issue on Recent Advances in Theory, Methodology and Applications of Imbalanced Learning.
- **Yones, C.**, Macchiaroli, N. Kamenetzky, L., Stegmayer, G., y Milone, D. (2018). Hextractor: an R package for automatic extraction of hairpin sequences in genome-wide data. Trabajo enviado a Bioinformatics.

Capítulo de libro

- Stegmayer, G., **Yones, C.**, Kamenetzky, L., Macchiaroli, N., & Milone, D. H. (2017). Computational Prediction of Novel miRNAs from Genome-Wide Data. In *Functional Genomics* (pp. 29-37). Humana Press, New York, NY.

Trabajos en eventos científicos

- **Yones, C.**, Stegmayer, G., Kamenetzky, L., Milone, D. (2016). Automatic extraction of hairpin sequences from genome-wide data. ISCB Latin America 2016
- **Yones, C.**, Stegmayer, G., Kamenetzky, L., Milone, D. (2015). miRNAfe a tool for feature extraction in pre-miRNA prediction. VI Congreso Argentino de Bioinformática y Biología Computacional (CAB2C)
- **Yones, C.**, Stegmayer, G., Milone, D. (2015). miRNAAss a semi-supervised approach for microRNA prediction. VI Congreso Argentino de Bioinformática y Biología Computacional (CAB2C)

Bibliografía

- An, J., Lai, J., Lehman, M. L., and Nelson, C. C. (2013). miRDeep*: an integrated application tool for miRNA identification from RNA sequencing data. *Nucleic acids research*, **41**(2), 727–737.
- Batuwita, R. and Palade, V. (2009). microPred: effective classification of pre-miRNAs for human miRNA gene prediction. *Bioinformatics*, **25**(8), 989–995.
- 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.
- Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern recognition*, **30**(7), 1145–1159.
- Chapelle, O., Schölkopf, B., and Zien, A. (2006). *Semi-supervised Learning*. Adaptive computation and machine learning. MIT Press.
- Ding, J., Zhou, S., and Guan, J. (2010). MiRenSVM: towards better prediction of microRNA precursors using an ensemble SVM classifier with multi-loop features. *BMC bioinformatics*, **11**(Suppl 11), S11.
- Durbin, R. and Rice, P. (1999). Einverted: finds dna inverted repeats (emboss). *European Bioinformatics Institute, Cambridge*.
- Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, **32**(200), 675–701.
- Gander, W., Golub, G. H., and von Matt, U. (1989). A constrained eigenvalue problem. *Linear Algebra and its applications*, **114**, 815–839.
- Gudyś, A., Szcześniak, M. W., Sikora, M., and Makałowska, I. (2013). HuntMi: an efficient and taxon-specific approach in pre-miRNA identification. *BMC bioinformatics*, **14**(1), 83.
- Guo, X., Yin, Y., Dong, C., Yang, G., and Zhou, G. (2008). On the class imbalance problem. In *Natural Computation, 2008. ICNC'08. Fourth International Conference on*, volume 4, pages 192–201. IEEE.
- Hackenberg, M., Sturm, M., and Langenberger, D. (2009). miRAnalyzer: a microRNA detection and analysis tool for next-generation sequencing experiments. *Nucleic Acids Research*, **37**, 68–76.
- Hertel, J. and Stadler, P. (2006). Hairpins in a haystack: recognizing microRNA precursors in comparative genomics data. *Bioinformatics*, **22** (14), e197–e202.
- Huang, T.-H., Fan, B., Rothschild, M. F., Hu, Z.-L., Li, K., and Zhao, S.-H. (2007). Mirfinder: an improved approach and software implementation for genome-wide fast microrna precursor scans. *BMC bioinformatics*, **8**(1), 341.
- Jha, A., Chauhan, R., Mehra, M., Singh, H. R., and Shankar, R. (2012). mir-bag: bagging based identification of microrna precursors. *PLoS One*, **7**(9), e45782.

- Jiang, P., Wu, H., Wang, W., Ma, W., Sun, X., and Lu, Z. (2007). MiPred: classification of real and pseudo microRNA precursors using random forest prediction model with combined features. *Nucleic acids research*, **35**(suppl 2), W339–W344.
- Joachims, T. *et al.* (2003). Transductive learning via spectral graph partitioning. In *ICML*, volume 3, pages 290–297.
- Kadri, S., Hinman, V., and Benos, P. V. (2009). Hhmmir: efficient de novo prediction of micrornas using hierarchical hidden markov models. *BMC bioinformatics*, **10**(1), S35.
- Kleftogiannis, D., Korfiati, A., Theofilatos, K., Likothanassis, S., Tsakalidis, A., and Mavroudi, S. (2013). Where we stand, where we are moving: surveying computational techniques for identifying miRNA genes and uncovering their regulatory role. *Journal of biomedical informatics*, **46**(3), 563–573.
- Kononenko, I. (1994). Estimating attributes: analysis and extensions of RELIEF. In *Machine Learning: ECML-94*, pages 171–182. Springer.
- Kozomara, A. and Griffiths-Jones, S. (2014). mirbase: annotating high confidence micrornas using deep sequencing data. *Nucleic Acids Research*, **42**, D68–D73.
- Lecellier, C., Dunoyer, P., Arar, K., Lehmann-Che, J., Eyquem, S., Himber, C., Saib, A., and Voinnet, O. (2005). A cellular microRNA mediates antiviral defense in human cells. *Science*, **308**, 557–560.
- Lim, L. P., Lau, N. C., Weinstein, E. G., Abdelhakim, A., Yekta, S., Rhoades, M. W., Burge, C. B., and Bartel, D. P. (2003). The microRNAs of *Caenorhabditis elegans*. *Genes & development*, **17**(8), 991–1008.
- Liu, Q. and Chen, Y.-Q. (2010). A new mechanism in plant engineering: the potential roles of micrornas in molecular breeding for crop improvement. *Biotechnology advances*, **28**(3), 301–307.
- Liu, X., He, S., Skoerboe, G., Gong, F., and Chen, R. (2012). Integrated sequence-structure motifs suffice to identify microrna precursors. *PloS one*, **7**(3), e32797.
- Lopes, I. d. O., Schliep, A., and de Carvalho, A. C. d. L. (2014). The discriminant power of RNA features for pre-miRNA recognition. *BMC Bioinformatics*, **15**(1), 124.
- Lopes, I. d. O., Alexander, S., and de LF de Carvalho André, P. (2016). Automatic learning of pre-miRNAs from different species. *BMC Bioinformatics*, **17**(1), 224.
- McCaskill, J. (1990). The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, **29**, 1105–1119.
- Mease, D., Wyner, A. J., and Buja, A. (2007). Boosted classification trees and class probability/quantile estimation. *The Journal of Machine Learning Research*, **8**, 409–439.
- Nemenyi, P. (1962). Distribution-free multiple comparisons. In *Biometrics*, volume 18, page 263. INTERNATIONAL BIOMETRIC SOC 1441 I ST, NW, SUITE 700, WASHINGTON, DC 20005-2210.
- Ng, K. L. S. and Mishra, S. K. (2007). De novo SVM classification of precursor microRNAs from genomic pseudo hairpins using global and intrinsic folding measures. *Bioinformatics*, **23**(11), 1321–1330.
- Olson, S. A. (2002). Emboss opens up sequence analysis. *Briefings in bioinformatics*, **3**(1), 87–91.
- Peace, R. J., Biggar, K. K., Storey, K. B., and Green, J. R. (2015). A framework for improving microRNA prediction in non-human genomes. *Nucleic acids research*, page gkv698.
- Rosenzvit, M., Cucher, M., Kamenetzky, L., Macchiaroli, N., Prada, L., and Camicia, F. (2013). MicroRNAs in endoparasites. *Nova Science Publishers*, pages 65–92.

- Russell, S. J. and Norvig, P. (2016). *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.
- Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, **3**(3), 210–229.
- Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, **22**(8), 888–905.
- Stegmayer, G., Di Persia, L., Rubiolo, M., Gerard, M., Pividori, M., Yones, C., Bugnon, L., Rodriguez, T., Raad, J., and Milone, D. (2018). Predicting novel microrna: a comprehensive comparison of machine learning approaches. *Briefings in bioinformatics*. doi: 10.1093/bib/bby037.
- Tempel, S., Zerath, B., Zehraoui, F., Tahiri, F., et al. (2015). miRBoost: boosting support vector machines for microRNA precursor classification. *RNA*, **21**(5), 775–785.
- Terai, G., Komori, T., Asai, K., and Kin, T. (2007). miRRim: a novel system to find conserved miRNAs with high sensitivity and specificity. *Rna*, **13**(12), 2081–2090.
- Tsetsarkin, K. A., Liu, G., Volkova, E., and Pletnev, A. G. (2017). Synergistic internal ribosome entry site/microrna-based approach for flavivirus attenuation and live vaccine development. *MBio*, **8**(2), e02326–16.
- Wei, L., Liao, M., Gao, Y., Ji, R., He, Z., and Zou, Q. (2014). Improved and promising identification of human microRNAs by incorporating a high-quality negative set. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, **11**(1), 192–201.
- Wenyuan, L., Jing, M., Changwu, W., Baowen, W., and Yongqiang, L. (2013). The training set selection methods of microRNA precursors prediction based on machine learning approaches. In *Intelligent System Design and Engineering Applications (ISDEA), 2013 Third International Conference on*, pages 1566–1569. IEEE.
- Wettschereck, D., Aha, D. W., and Mohri, T. (1997). A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, **11**(1-5), 273–314.
- Wu, Y., Wei, B., Liu, H., Li, T., and Rayner, S. (2011). Mirpara: a svm-based software tool for prediction of most probable microrna coding regions in genome scale sequences. *BMC bioinformatics*, **12**(1), 107.
- Xu, Y., Zhou, X., and Zhang, W. (2008). MicroRNA prediction with a novel ranking algorithm based on random walks. *Bioinformatics*, **24**(13), i50–i58.
- Xuan, P., Guo, M., Wang, J., Wang, C., Liu, X., and Liu, Y. (2011a). Genetic algorithm-based efficient feature selection for classification of pre-miRNAs. *Genet. Mol. Res.*, **10** (2), 588–603.
- Xuan, P., Guo, M., Liu, X., Huang, Y., Li, W., and Huang, Y. (2011b). Plantmirnapred: efficient classification of real and pseudo plant pre-mirnas. *Bioinformatics*, **27**(10), 1368–1376.
- Xue, C., Li, F., He, T., Liu, G.-P., Li, Y., and Zhang, X. (2005). Classification of real and pseudo microrna precursors using local structure-sequence features and support vector machine. *BMC bioinformatics*, **6**(1), 310.
- Yousef, M., Nebozhyn, M., Shatkay, H., Kanterakis, S., Showe, L., and Showe, M. (2006). Combining multi-species genomic data for microRNA identification using a naive bayes classifier. *Bioinformatics*, **22** (11), 1325–1334.
- Yu, T., Li, J., Yan, M., Liu, L., Lin, H., Zhao, F., Sun, L., Zhang, Y., Cui, Y., Zhang, F., et al. (2015). Microrna-193a-3p and-5p suppress the metastasis of human non-small-cell lung cancer by downregulating the erbB4/pik3r3/mtor/s6k2 signaling pathway. *Oncogene*, **34**(4), 413.

- Zuker, M. and Stiegler, P. (1981). Optimal computer folding of large rna sequences using thermodynamics and auxiliary information. *Nucleic acids research*, **9**(1), 133–148.

Apéndices

Contribuciones

HextractoR: an R package for automatic extraction of hairpins from genome-wide data

En este trabajo se publicó la herramienta de extracción de secuencias con estructura tipo tallo-horquilla de genomas completos. Esta publicación corresponde con la primera etapa de la metodología desarrollada en la tesis. En este trabajo me encargué de la revisión del estado del arte, del diseño y desarrollo de los algoritmos que componen la herramienta, de la validación y prueba de esta y de la escritura del manuscrito.

miRNAfe: a comprehensive tool for feature extraction in microRNA prediction

En este trabajo se publicó la herramienta de extracción de características de secuencias tipo tallo-horquilla. Esta publicación corresponde con la segunda etapa de la metodología desarrollada en la tesis. En este trabajo me encargué de la revisión del estado del arte, del diseño y desarrollo de la biblioteca, de la validación de los algoritmos de extracción de características, de la implementación de la interfaz web y de la escritura del manuscrito.

Genome-wide pre-miRNA discovery from few labeled examples

En este trabajo se presentó el método semi-supervisado de predicción de microRNA en genoma completo. Esta publicación corresponde con la tercera etapa de la metodología desarrollada en la tesis. En este trabajo mi contribución fue en el desarrollo de la idea, la ejecución de los experimentos y la redacción del manuscrito.

HextractoR: an R package for automatic extraction of hairpins from genome-wide data

HextractoR: an R package for automatic extraction of hairpins from genome-wide data

Cristian A. Yones^{*1}, Natalia Macchiaroli², Laura Kamenetzky², Georgina Stegmayer¹, and Diego H. Milone¹

¹Research Institute for Signals, Systems and Computational Intelligence, sinc(i), FICH-UNL, CONICET, Ciudad Universitaria UNL, (3000) Santa Fe, Argentina.

²Instituto de Investigaciones en Microbiología y Parasitología Médica (UBA), CONICET, Paraguay 2155, piso 13 (1121), Buenos Aires, Argentina.

Abstract

Summary: Extracting stem-loop sequences (hairpins) from genome-wide data is very important nowadays for some data mining tasks in bioinformatics. The genome pre-processing is very important because it has a strong influence on the later steps and the final results. For example, for novel miRNA prediction, all well-known hairpins must be properly located. Although there are some scripts that can be adapted and put together to achieve this task, they are outdated, none of them guarantees finding correspondence to well-known structures in the genome under analysis, and they do not take advantage of the latest advances in secondary structure prediction. We present here HextractoR, an R package for automatic extraction of hairpins from genome-wide data. HextractoR makes an exhaustive and smart analysis of the genome in order to obtain a very good set of short sequences for further processing. Moreover, genomes can be processed in parallel and with low memory requirements. Results obtained showed that HextractoR has effectively outperformed other methods.

Availability: HextractoR it is freely available at CRAN and <https://sourceforge.net/projects/sourcesinc/files>

Contact: cyones@sinc.unl.edu.ar

1 Introduction

Extracting stem-loop sequences (hairpins) from genome-wide data is very important for some data mining tasks in bioinformatics such as the computational prediction of pre-microRNAs (pre-miRNAs) with machine learning. In most works (Xue *et al.*, 2005; Gudys *et al.*, 2013; Demirci *et al.*, 2017; Stegmayer *et al.*, 2018) the datasets used to tests the prediction methods are manually built, using several interconnected tools. This has the obvious disadvantage of requiring a not negligible manual amount of work. Moreover, the process has a great impact in the prediction task afterwards. If some stem-loops are not correctly identified and extracted from the genome, the

^{*}cyones@sinc.unl.edu.ar

prediction method will not be able to detect the corresponding sequence. If they are detected but incorrectly trimmed (longer or shorter than a corresponding pre-miRNA) the features extracted from these sequences can vary a lot, making machine learning prediction methods to generate incorrect predictions. There are other tools to extract hairpins (Yang and Li, 2011; Friedländer *et al.*, 2008) but they use RNAseq data and they are not designed to extract all hairpins, only the ones that match with a significant number of reads. Finally, since in most works this first stage of stem-loop extraction is performed manually by combining several tools, it is very difficult, or even impossible, to reproduce the results. This makes that experiments of most pre-miRNA prediction methods published cannot be accurately reproduced, and also that the users of those tools cannot obtain the same prediction rates published. With HextractoR, besides providing a unique tool to simplify this stage of pre-miRNA prediction, a standardized way to perform this important preprocessing task is proposed. After hairpins extraction, the miRNAfe tool (Yones *et al.*, 2015) can be used, which combines all the features previously described for pre-miRNAs in a single tool. It is actually being used in most recent prediction models (Yones *et al.*, 2017; Acar *et al.*, 2018). HextractoR helps to standardize and simplify the stem-loop extraction stage, making future prediction methods more easy to use and their experiments fully reproducible.

2 HextractoR pipeline

HextractoR predicts the secondary structure of several overlapped segments, with a longer length than the mean length of the sequences of interest for the species under processing, ensuring that no one is lost nor inappropriately cut. The length of the cutting window can be configured to define the maximum size that the stems found will have (smaller stems will also be found). Optionally, FASTA files containing known sequences can be loaded and used to split the output stem-loops into several FASTA output files, according to their type. If no filter file is provided HextractoR generates just one FASTA file with all hairpins. If two filter files are provided, for example, with well-known pre-miRNAs and other with known non-miRNA sequences, HextractoR generates three FASTA files: one for each filtering file passed to HextractoR with the sequences that match according to BLAST (Altschul *et al.*, 1990) with known sequences; and another one with the stem-loops that did not match. The processing steps of HextractoR (see Figure 1) are explained in detail in the next sections.

Intelligent windowing of the whole genome. HextractoR starts by cutting the complete genome into overlapping windows of a large length (~ 500 nt). The window must be long enough in order to correctly capture a complete hairpin, but also to take into account the neighborhood of any possible hairpin when estimating the secondary structure. This is very important since the results of estimating a secondary structure can be greatly affected by the neighborhood of the sequences.

Folding and splitting. Prediction of the secondary structures of the sequences obtained when folding. The minimum free energy algorithm (Zuker and Stiegler, 1981) of RNAfold is used. Since the windows used are relatively long, the structures found usually have multiple loops. Therefore, they have to be split into several hairpin-type structures. Those hairpins that do not exceed a minimum length and level of pairing are eliminated.

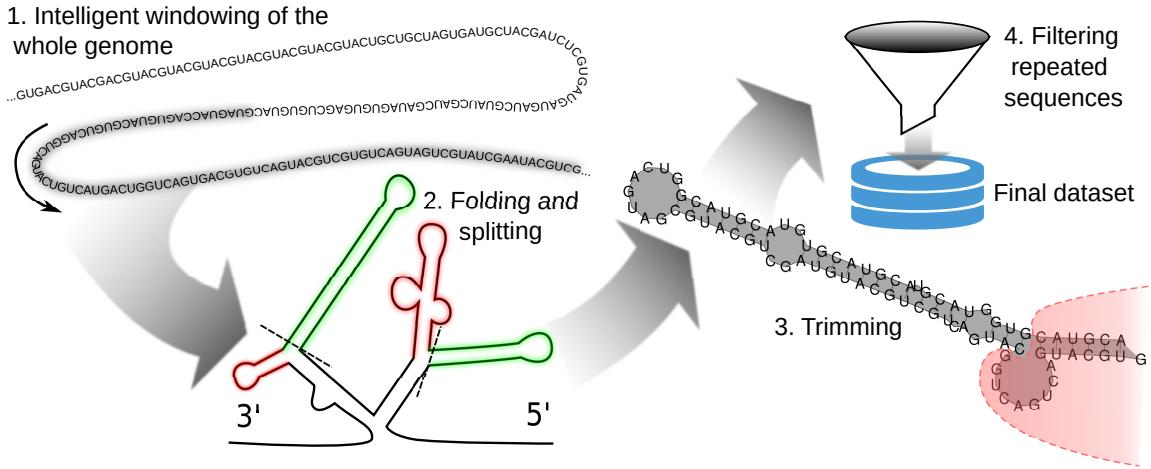


Figure 1: The HextractoR pipeline.

Trimming. Certain heuristics can be used to obtain sequences with lengths and stability properties similar to those of a well-known pre-miRNA. These rules optimize the Minimum Free Energy normalized by the sequence length (NMFE). Although optimum cutting points can be found by re-estimating the secondary structure for all possible cuts, a set of rules provide more flexibility and accelerate the process. First, the sequence must exceed a minimum length, pre-defined according to the species under study. In this way, it can be ensured that the secondary structure has sufficient length to be a pre-miRNA of the species under analysis. Secondly, the cuts are made in the first unpaired nucleotide of an internal loop or bulge of the secondary structure (starting from the main loop). To choose the bulge/loop where to cut, a score is assigned to each imperfection as $S(1 - D/L)^2$, where S is the imperfection length, D is the distance to the main loop and L is the stem length (from the main loop to the last paired nucleotide). If the imperfections in the secondary structure are large, it is likely that cutting the sequence at those points will result in a structure with lower NMFE. Moreover, the smaller the length of the sequence (independently of the pairing), the higher the NMFE. Therefore, a loop/bulge closer to the main loop is preferred.

Filtering repeated sequences. Repeated sequences are eliminated to avoid extra computational cost. These repeated sequences might also disturb the results of the prediction algorithms. Repetitions may appear due to the overlapping in windowing. These repeated sequences appear consecutively and they are almost identical sequences. To eliminate them, a comparison between each sequence and the last extracted sequence is made. If one of the sequences contains the other one, the shorter one can be discarded.

3 Complete genomes results

To test the R package, several genomes were processed and the results were compared to those obtained from using some mirCheck scripts (Jones-Rhoades, 2010).

These scripts were the only tool that we have found to extract stem-loop sequences from genome-

Table 1: Number of stem-loops and pre-miRNA found with each tool

Species	Extracted hairpins		pre-miRNAs found		known pre-miRNAs
	mirCheck	HExtractor	mirCheck	HExtractor	
<i>A. gambiae</i>	1,410,532	4,276,543	92.42 %	100.00 %	66
<i>C. elegans</i>	875,588	1,739,124	90.00 %	99.60 %	250
<i>D. rerio</i>	11,028,128	23,214,338	93.06 %	99.42 %	346
<i>E. multilocularis</i>	509,530	1,898,911	81.81 %	100.00 %	22
<i>A. thaliana</i>	874,320	1,357,455	91.69 %	94.77 %	325
<i>H. sapiens</i>	18,654,426	48,206,494	85.38 %	97.45 %	1881

wide data. Table 1 shows the results for six selected species. The number of well-known pre-miRNAs (according to miRBase v21(Griffiths-Jones *et al.*, 2006)) for all species is reported in the fourth column of the table. The proportion of known pre-miRNAs found is shown in the last two columns. It can be clearly seen that, in all cases, the performance of HextractoR is superior to mirCheck.

4 Conclusion

We have developed a simple and integrated tool, the R package HextractoR, that automatically extracts and folds all possible hairpin sequences from genome-wide data. The genomes can be processed in parallel and with low memory requirements since it can automatically split large multi-FASTA files. The proposed computational method, takes advantage of the latest developments in secondary structure prediction. Results obtained showed that HextractoR has effectively outperformed other methods.

Funding

This study was supported by CONICET [PIP 117], UNL [CAI+D 2016 082], and ANPCyT [PICT 2014 2627]. Conflict of Interest: none declared.

References

- Acar, İ., Saçar Demirci, M., Groß, U., and Allmer, J. (2018). The expressed microRNA-mRNA interactions of toxoplasma gondii. *Frontiers in microbiology*, **8**, 2630.
- Altschul, S., Gish, W., Miller, W., Myers, E., and Lipman, D. (1990). Basic local alignment search tool. *Journal of molecular biology*, **215**(3), 403–410.
- Demirci, M., Baumbach, J., and Allmer, J. (2017). On the performance of pre-microRNA detection algorithms. *Nature communications*, **8**(1), 330.

- Friedländer, M., Chen, W., Adamidi, C., Maaskola, J., Einspanier, R., Knespel, S., and Rajewsky, N. (2008). Discovering micrornas from deep sequencing data using mirdeep. *Nature biotechnology*, **26**(4), 407.
- Griffiths-Jones, S., Grocock, R., Van Dongen, S., Bateman, A., and Enright, A. (2006). mirbase: microRNA sequences, targets and gene nomenclature. *Nucleic acids research*, **34**(suppl_1), D140–D144.
- Gudyś, A., Szcześniak, M., Sikora, M., and Makałowska, I. (2013). Huntmi: an efficient and taxon-specific approach in pre-mirna identification. *BMC bioinformatics*, **14**(1), 83.
- Jones-Rhoades, M. (2010). *Prediction of plant miRNA genes*. Springer.
- Stegmayer, G., Di Persia, L., Rubiolo, M., Gerard, M., Pividori, M., Yones, C., Bugnon, L., Rodriguez, T., Raad, J., and Milone, D. (2018). Predicting novel microrna: a comprehensive comparison of machine learning approaches. *Briefings in bioinformatics*. doi: 10.1093/bib/bby037.
- Xue, C., Li, F., He, T., Liu, G., Li, Y., and Zhang, X. (2005). Classification of real and pseudo microRNA precursors using local structure-sequence features and support vector machine. *BMC bioinformatics*, **6**(1), 310.
- Yang, X. and Li, L. (2011). mirdeep-p: a computational tool for analyzing the microrna transcriptome in plants. *Bioinformatics*, **27**(18), 2614–2615.
- Yones, C., Stegmayer, G., Kamenetzky, L., and Milone, D. (2015). mirnaf: a comprehensive tool for feature extraction in microRNA prediction. *Biosystems*, **138**, 1–5.
- Yones, C., Stegmayer, G., and Milone, D. (2017). Genome-wide pre-mirna discovery from few labeled examples. *Bioinformatics*, **34**(4), 541–549.
- Zuker, M. and Stiegler, P. (1981). Optimal computer folding of large rna sequences using thermodynamics and auxiliary information. *Nucleic acids research*, **9**(1), 133–148.

miRNAfe: a comprehensive tool for feature extraction in microRNA prediction

miRNAfe: a comprehensive tool for feature extraction in microRNA prediction

Cristian A. Yones^{*1}, Georgina Stegmayer¹, Laura Kamenetzky², and Diego H. Milone¹

¹Research Institute for Signals, Systems and Computational Intelligence, sinc(i), FICH-UNL, CONICET,
Ciudad Universitaria UNL, (3000) Santa Fe, Argentina.

²Instituto de Investigaciones en Microbiología y Parasitología Médica (UBA), CONICET, Paraguay 2155,
piso 13 (1121), Buenos Aires, Argentina.

Abstract

miRNAfe is a comprehensive tool to extract features from RNA sequences. It is freely available as a web service, allowing a single access point to almost all state-of-the-art feature extraction methods used today in a variety of works from different authors. It has a very simple user interface, where the user only needs to load a file containing the input sequences and select the features to extract. As a result, the user obtains a text file with the features extracted, which can be used to analyze the sequences or as input to a miRNA prediction software.

The tool can calculate up to 80 features where many of them are multidimensional arrays. In order to simplify the web interface, the features have been divided into six pre-defined groups, each one providing information about: primary sequence, secondary structure, thermodynamic stability, statistical stability, conservation between genomes of different species and substrings analysis of the sequences. Additionally, pre-trained classifiers are provided for prediction in different species. All algorithms to extract the features have been validated, comparing the results with the ones obtained from software of the original authors.

The source code is freely available for academic use under GPL license at <http://sourceforge.net/projects/sourcesinc/files/mirnafe/0.90/>. A user-friendly access is provided as web interface at <http://fich.unl.edu.ar/sinc/web-demo/mirnafe/>. A more configurable web interface can be accessed at <http://fich.unl.edu.ar/sinc/web-demo/mirnafe-full/>.

keywords: *microRNA, Feature extraction, Web tool*

1 Introduction

MicroRNAs (miRNA) are a group of short (~ 22 nucleotides) non-coding RNA which can play important roles in gene regulation by targeting mRNAs for cleavage or translational repression (Lamers *et al.*, 2014). Precursors of miRNA (pre-miRNA) are characterized by their hairpins structure. However, a large amount of similar sequences can be folded into this kind of structure in many genomes.

^{*}cyones@sinc.unl.edu.ar

In order to predict miRNAs, a large number of tools have been developed in the last years (Kleftogiannis *et al.*, 2013). The first step is to extract features from sequences and then use classifiers to predict which sequences are likely to contain a miRNA. The feature extraction step is very important for the whole process, in order to achieve high rates of true positives predictions (Zhang *et al.*, 2010). Numerous features can be extracted from the primary sequence and its corresponding secondary structure. A typical example of this kind of features is the *triplets* representation (Xue *et al.*, 2005), which considers the structural composition of three adjacent nucleotides and the middle base to build a vector with 32 elements. Other examples are the number of internal loops and their length (Yousef *et al.*, 2006), the z-score of the minimum free energy (Hertel and Stadler, 2006) and the dinucleotide proportion (Rukshan and Vasile, 2009). The amount of features that can be extracted is very large and there are many different tools that partially achieve this task. They are coded in different programming languages and have different access modes (web, command line, etc.). Besides, several tools are proprietary software and the source code is not even available¹. These are important issues that hinder their use.

We have developed the miRNAfe tool that implements almost all existing state-of-the-art feature extraction processes used for miRNA prediction nowadays (Li *et al.*, 2010). It can extract the features used by the most cited miRNA classifiers, such as Triplet-SVM (Xue *et al.*, 2005), RNAMicro (Hertel and Stadler, 2006), BayesMiRNAfind (Yousef *et al.*, 2006), MiRFinder (Huang *et al.*, 2007), MiPred (Jiang *et al.*, 2007), miRRim (Goro *et al.*, 2007), microPred (Rukshan and Vasile, 2009), miRAnalyzer (Hackenberg *et al.*, 2009), MiRenSVM (Jiandong *et al.*, 2010) and miPredGA (Xuan *et al.*, 2011). We have developed an easy to use web interface that allows a single and simplified access point to all the functions of the toolbox, and a set of pre-trained classifiers that can be used to test the prediction power of the feature sets. We provide here a comprehensive open-source solution, with free access to all features for academic use.

2 Provided features

The tool implements up to 80 features, where many of them return arrays. All of these features have been proposed in literature over the past 10 years. The features are divided into six pre-defined groups according to the kind of information that must be extracted from the sequence. A brief explanation of each group is provided in the next lines. For a more detailed explanation of all the features provided and their sources, see the supplementary material.

2.1 Sequence

These are the simplest features and represent information from the primary sequence. MiRNAfe can extract a total of 5 features in this group: sequence length (ℓ), proportion of each base in the sequence, proportion of dinucleotides, content of guanine and cytosine and guanine-cytosine ratio. The last two features are defined as:

$$G + C_{content} = \frac{G + C}{G + C + A + U}, \quad (1)$$

$$GC_{ratio} = \frac{G}{C}, \quad (2)$$

¹<http://www.insybio.com/pages/ncrnaseq>

where G , C , A and U represent the quantity of each base found in the sequence (Hertel and Stadler, 2006). All these features form a vector of 23 elements, composed by: the 4 base proportions, the 16 dinucleotide proportions, sequence length, $G + C_{content}$ and GC_{ratio} . Although these features are quite simple, they have shown a high discriminative power (Rukshan and Vasile, 2009), and thus are used in most of the state-of-the-art prediction software.

2.2 Secondary structure

These features represent information from the secondary structure and they are the most numerous group. The most used feature of this group is the triplets proportion (Xue *et al.*, 2005). A triplet is an element formed with the structure state (paired or not paired) of three adjacent nucleotides and the base at the middle. An example of a triplet element is “.(A”, where the parenthesis represents a paired nucleotide, a dot a not paired one, and the letter is the base of the nucleotide in the middle. As there are 2 possible states for a nucleotide and 4 different bases, 32 triplets can be formed (4×2^3). The number of occurrences of each triplet element in the sequence is counted and normalized to produce a 32-dimensional feature vector. A similar approach to the triplets was used by Huang *et al.* (2007), which proposed another representation for the secondary structure. First of all, five symbols are defined to indicate the status of each base pair in the stem: “=”, “.”, “-”, “.” and “ \wedge ”. Each of them corresponds to the status of match, mismatch, deletion, insertion in the interior loop, and insertion in the bulged loop, respectively. Then, by taking two adjacent symbols, 14 possible combinations can be formed, each one having a special meaning. For example: “= -”, “= .”, and “= :” represent the boundary of the stem/loop, and “: \wedge ” represents that the loop is asymmetric. The frequency of each combination is used as a feature vector. This representation is also used to calculate four more features: $pMatch$, $pMismatch$, pDI and $pBulge$. These features are calculated over putative mature miRNA, selected as the 22 nucleotide region where base-pairing is maximum. They represents the base pairing frequency, the non-pairing frequency, the deletion and insertion frequencies and the symmetry of the bulged loops, respectively.

Another kind of features is related to the stems, which are structural motifs containing more than three contiguous base pairs (Ng and Mishra, 2007). These features are the number of stems, the proportion of each possible base pair per stem, average base pair number per stem and length of the longest stem. The rest of features are the stem region (the stem part of the stem-loop) length, terminal loop length, bulges number, loops number, longest loop length, asymmetric and symmetric loops number, nucleotides in symmetric and asymmetric loops, longest symmetric region, average length of symmetric loops, average length of asymmetric loops, number of bulges and loops of length 1, 2, ..., 7 and greater, base pair number, adjusted base pair propensity, base pair proportion and $G + C_{content}$ in the terminal loop (Lopes *et al.*, 2014). Finally, miRNAfe can calculate reads count from RNAseq data. This feature needs the user to provide an extra file with reads, which miRNAfe aligns with the analyzed sequences and counts the corresponding matches. For a full description of each feature see the supplementary material.

2.3 Thermodynamics stability

The features in this group are related to the thermodynamics stability of a sequence. The mostly used feature is the minimum free energy (MFE): the estimated energy that one sequence frees when folded into the most stable secondary structure (Zuker and Stiegler, 1981). The ensemble free energy (EFE) has a similar meaning and it is obtained with the algorithm from McCaskill (1990).

Other features of this group are calculated as combinations of those values. For example, the MFE index 1 ($MFEI_1$) is the ratio between the minimum free energy and the $G + C_{content}$ defined in 1. Similarly, miRNAfe can calculate $MFE - EFE$ difference, adjusted MFE , $MFEI_2$, $MFEI_3$ and $MFEI_4$ (Rukshan and Vasile, 2009). There are also some features that use information theoretic approaches to estimate the confidence of the predicted secondary structure, such as the adjusted Shannon entropy of the pairing probabilities (Ng and Mishra, 2007), defined as

$$dQ = \frac{1}{\ell} \sum_{i < j} p_{ij} \log_2 p_{ij}, \quad (3)$$

where p_{ij} is the probability that the nucleotide i forms a pair with the nucleotide j and ℓ is the sequence length. The base pair probabilities are calculated with the algorithm from McCaskill (1990). Another example is the adjusted base pair distance, defined as

$$dD = \frac{1}{\ell} \sum_{i < j} p_{ij}(1 - p_{ij}). \quad (4)$$

Additionally, in this group miRNAfe can calculate the ensemble frequency, set diversity, stem 3' and 5' potential, and loop potential (Terai *et al.*, 2007). There are 15 features in this group, which are described in more detail in the supplementary material.

2.4 Statistical stability

It is well-known that precursors containing a miRNA are more stable than random sequences. The features in this group are calculated as the standard score of any feature related to stability. To calculate this score, a random population of sequences has to be generated swapping the bases of the analyzed sequence. This way, the artificially generated sequences conserve the nucleotide proportions or even the dinucleotide proportion if some swaps are restricted (the tool has an option to choose which swap method to use). For each generated sequence, the stability can be measured with the z-score (Bonnet *et al.*, 2004), defined as

$$z = \frac{x - \mu}{\sigma}, \quad (5)$$

where x is the original value of the feature, μ is the mean and σ is the standard deviation of the randomly generated population of sequences. This score represents how many standard deviations a value is above the population mean. Thus, a negative z-score indicates a sequence that is statistically more stable than the population mean. Another statistic used to measure the stability of the sequence in comparison with random sequences is the p-value. It is calculated as the proportion of random sequences that are more stable than the analyzed sequence. Thus, a low p-value indicates that the analyzed sequence is one of the most stable of all sequences generated with that nucleotide/dinucleotide proportion. The stabilities measures that can be normalized with z-score are: MFE (named $zMFE$), EFE ($zEFE$), adjusted MFE (zG), Shannon's entropy (zQ), base pair propensity (zP) (Ng and Mishra, 2007) and base pair distance (zD) (Jiandong *et al.*, 2010). The p-value can be used to normalize the MFE ($pMFE$) (Bonnet *et al.*, 2004) and the EFE ($pEFE$) (Jiandong *et al.*, 2010). Although z-score and p-value are alternative statistics for these features, they are often used together in prediction since they can take very different values (Jiandong *et al.*, 2010).

In summary, miRNAfe can calculate 8 features in this group. In the full version, the user can specify the shuffling method (preserving nucleotide or dinucleotide composition) and the number of random sequences generated. For the user-friendly web-interface, these parameters are set by default to 1000 random sequences and preservation of dinucleotide composition.

2.5 Phylogenetic conservation

When a portion of the genome is conserved between related species, it is highly likely to have an important role in the genome. The features of this group measure the level of conservation between sequences of phylogenetically related species. All the features are calculated over alignments of two or more sequences that the user must provide. Some features do not only take into account the conservation level, but also the thermodynamic stability. The features in this group are: the mutation frequency (Huang *et al.*, 2007), which is the proportion of bases that differ from one sequence to another and it is applicable only to a pair of sequences; the column entropy of the 5' arm, 3' arm, loop region and minimum entropy, which is the Shannon entropy calculated over a region of 21 nucleotides (Hertel and Stadler, 2006); the number of differences in the secondary structure divided by the number of differences between sequences (Huang *et al.*, 2007); the average *MFE*; the *MFE* difference between two aligned sequences, divided by the number of differences between the sequences(Huang *et al.*, 2007); average *dG*; average *MFEI*₁; free energy of the consensus secondary structure; conservation of the 3' arm and conservation of the 5' arm; and finally, the conservation score. This is the most complex feature to obtain (Goro *et al.*, 2007), because is calculated using two Markov processes, one that moves in the time dimension (over the branches of the evolution tree), and the other in space dimension (over the sequence). A total of 14 features can be extracted in this group, which are described in detail in the supplementary material.

2.6 22-nt substring analysis

These features are calculated over all 22 nt substrings within a given sequence. They are based on the fact that if one sequence is a pre-miRNA, one of the analyzed substring has to be the mature miRNA and the features calculated must capture its particularities. As a result, an array with length $n = \ell - 22$ is obtained, where the i -th element represents the value of the calculated feature over the substring that starts at the base i . MiRNAfe can extract the following 5 features in this group: the base-pairing probability in the substring (Lim *et al.*, 2003), which is the sum of the base-pairing probability over the substring; the sum of not paired bases on the substring; the sum of the base-pairing probability on the secondary structure, without the probabilities of the nucleotides on the substring; the bulge symmetry, as the difference between the amount of not paired bases on each arm of the substring; and the distance from the substring to the terminal loop.

3 Implementation

MiRNAfe is composed by a set of Matlab functions which prepare the input sequences and implement the feature extraction processes. The source code is platform independent and provides functions that allow batch processing, saving of results and show reports with the extracted features. These functions can be installed in the user machine and used as any other Matlab toolbox. Thus the user is able to extract all the features present in miRNAfe and also make predictions. The toolbox has a main function that takes as parameters the path of the input fasta file and

a configuration file written in *yaml*², which is a human readable format that allows editing in a simple way on any text editor. This file contains folding options, alignment parameters to make phylogenetic related tests, a list of features to extract, post-process options such as normalization or sequence filtering by minimum free energy. The toolbox is very versatile and can be extended easily, only saving the function in the source folder and adding one line to the configuration file. Also, miRNAfe when used as a toolbox can train a classifier and optimize its parameters. In this case, the user only needs to provide positive and negative examples of miRNA sequences of a target specie or family. In the configuration file, options related to the SVM training and the parameters optimization stage can also be specified. The toolbox uses external well-known software to implement some standard processes, like folding or aligning sequences. The Vienna RNA³ package is used to fold the sequences and for alignments of sequences. Since the software RNAfold is used in almost all feature extraction process, miRNAfe shares its same restrictions about sequence lengths. Thus, a limit of 5000 nucleotides was imposed to avoid memory problems. For the phylogenetic related features, the software ClustalW⁴ is used to align sequences, Bowtie⁵ is used to align reads to sequences and PHAST⁶ is used to calculate the conservation score.

4 Web interface

To provide a more user-friendly access, we have developed a simplified and easy to use web interface using the tool provided by Stegmayer *et al.* (2015). It can be accessed at <http://fich.unl.edu.ar/sinc/web-demo/mirnafe/>. As it can be seen in Figure 1, the user must load a fasta file with the sequences to be analyzed. After that, he/she can select which group of features wants to extract by checking the corresponding checkbox. Then, a pre-trained classifier on the sequences under analysis can be used for prediction. A support vector-machine (SVM) is provided for classification since it is the most frequently used method for pre-miRNA prediction (Kleftogiannis *et al.*, 2013). Sequences from two genomes were used as training data, *Homo sapiens* and *Arabidopsis thaliana*. To create positive sets, all known pre-miRNAs from those species in miRBase release 21⁷ (Kozomara and Griffiths-Jones, 2014) were used. Negative sets were built by extracting random sequences from the genomes and mRNAs of these species. The sequence length distribution in the negative dataset was the same as in the corresponding positive one. The extracted sequences were filtered to preserve only sequences with minimum free energy below -0.05 (normalized to the sequence length) and proportion of paired bases in the stem above 0.15, similarly to Gudyś *et al.* (2013). Since the features of the substring group are arrays of variable length, they cannot be used for prediction directly in a standard SVM. Similarly, since the features related to conservation are calculated over several sequences altogether, they cannot be used directly for prediction with the pre-trained SVM.

Finally, when the feature extraction process is finished, the web interface provides links to download the output files in comma separated values (*csv*) format. This file format can be opened by any spreadsheet program and it is supported by almost all software used in bioinformatics. The features file contains, in each row, the name of the sequence analyzed, and in the first row the names of the corresponding features extracted. If the user has chosen to make a prediction, an extra

²<http://www.yaml.org/>

³<http://www.tbi.univie.ac.at/RNA/>

⁴<http://www.clustal.org/omega/>

⁵<http://bowtie-bio.sourceforge.net/index.shtml>

⁶<http://compgen.bscb.cornell.edu/phast/index.php>

⁷<http://www.mirbase.org/>

miRNAfe

A comprehensive tool of feature extraction for pre-miRNA prediction

Version: 0.9

Sequences: No file selected.

[Download sample data file](#)

Feature groups:

Sequence: Secondary structure:
Thermodynamic stability: Statistical stability:
22-nt substring analysis: Phylogenetic conservation:

Predictor for:

Features: [+](#)

Prediction: [+](#)

Log: [+](#)

Figure 1: Web interface of miRNAfe after analyzing some sample sequences

file can be downloaded as well, which lists all input sequences ordered from most to less probable miRNA, with a flag indicating if it was classified as miRNA (positive flag) or not (negative flag). Finally, a log file is also provided, which describes all the process stages, warnings and errors, if any.

5 Validation of the feature extraction processes

In order to validate all feature extraction scripts, several sequences were analyzed with miRNAfe and with the software of the original authors, and the outputs have been compared. The sequences used in this validation step were the well-known pre-miRNAs: ppa-mir-101, hsa-mir-34a, hsa-mir-7-1, hsa-let-7a-1, hsa-let-7a-3 and hsa-let-7b3. These sequences were selected randomly with the sole purpose of validating the feature extraction functions. The software used to validate the features extraction were MiRFinder (Huang *et al.*, 2007), miPred (Jiang *et al.*, 2007) and microPred (Rukshan and Vasile, 2009). MiRNAfe validates the extracted features by comparison with the reference software, when available. In most cases, however, the original scripts were not

available and reference results were obtained directly from data with already extracted features, and compared with the results of miRNAfe. In all tests performed, the results were always consistent to those of the original papers.

The validation process was automated with a script that is distributed together with the source code of the tool. This script prints on screen the results of each test, the reference value and the software that was used to obtain it. This can also be used to make tests and re-validate the results of each function after making improvements or alternative implementations of the algorithms. For the statistical features, where results are not deterministic, a confidence interval for the expected values was set according to the variance in each case. The validation error was calculated as $e = 100\|C - E\|_2/\|E\|_2$, where C is the result calculated by miRNAfe, E is the expected value (the one calculated with the software of the original author) and $\|\cdot\|_2$ is the norm 2. A list with the software used for comparisons and their corresponding references are provided in the supplementary material.

6 Acknowledgments

This work was supported by National Scientific and Technical Research Council [PIP 2013 117], National University of Litoral [CAI+D 2011 548] and Agencia Nacional de Promoción Científica y Tecnológica (ANPCyT) [PICT 2014 2627].

References

- 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.
- Goro, T., Takashi, K., Kiyoshi, A., and Taishin, K. (2007). mirrim: A novel system to find conserved miRNAs with high sensitivity and specificity. *RNA*, **13** (12), 2081–2090.
- Gudyś, A., Szcześniak, M. W., Sikora, M., and Makałowska, I. (2013). Huntmi: an efficient and taxon-specific approach in pre-mirna identification. *BMC bioinformatics*, **14**(1), 83.
- Hackenberg, M., Sturm, M., and Langenberger, D. (2009). miRanalyzer: a microRNA detection and analysis tool for next-generation sequencing experiments. *Nucleic Acids Research*, **37**, 68–76.
- Hertel, J. and Stadler, P. (2006). Hairpins in a haystack: recognizing microRNA precursors in comparative genomics data. *Bioinformatics*, **22** (14), e197–e202.
- Huang, T., Fan, B., Rothschild, M., Hu, Z., Li, K., and Zhao, S. (2007). MiRFinder: an improved approach and software implementation for genome-wide fast microRNA precursor scans. *BMC Bioinformatics*, **8**(1), 341.
- Jiandong, D., Shuigeng, Z., and Jihong, G. (2010). MirenSVM: towards better prediction of microRNA precursors using an ensemble SVM classifier with multi-loop features. *BMC Bioinformatics*, **11** (11), 11.

- Jiang, P., Wu, H., Wang, W., Ma, W., Sun, X., and Lu, Z. (2007). Mipred: classification of real and pseudo microRNA precursors using random forest prediction model with combined features. *Nucleic acids research*, **35**(suppl 2), W339–W344.
- Kleftogiannis, D., Korfiati, A., Theofilatos, K., Likothanassis, S., Tsakalidis, A., and Mavroudi, S. (2013). Where we stand, where we are moving: Surveying computational techniques for identifying mirna genes and uncovering their regulatory role. *Journal of biomedical informatics*, **46**(3), 563–573.
- Kozomara, A. and Griffiths-Jones, S. (2014). mirbase: annotating high confidence microRNAs using deep sequencing data. *Nucleic Acids Research*, **42**, D68–D73.
- Lamers, S. L., Fogel, G. B., Nolan, D. J., McGrath, M. S., and Salemi, M. (2014). Hiv-associated neuropathogenesis: A systems biology perspective for modeling and therapy. *Biosystems*, **119**, 53–61.
- Li, L., Xu, J., Yang, D., Tan, X., and Wang, H. (2010). Computational approaches for microRNA studies: a review. *Mamm Genome*, **21**(1), 1–12.
- Lim, L., Lau, N., Weinstein, E., Abdelhakim, A., Yekta, S., Rhoades, M., Burge, C., and Bartel, D. (2003). The microRNAs of *caenorhabditis elegans*. *Genes & development*, **17**(8), 991–1008.
- Lopes, I., Schliep, A., and de Carvalho, A. C. d. L. (2014). The discriminant power of rna features for pre-mirna recognition. *BMC bioinformatics*, **15**(1), 124.
- McCaskill, J. (1990). The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, **29**, 1105–1119.
- Ng, K. and Mishra, S. (2007). De novo SVM classification of precursor microRNAs from genomic pseudo hairpins using global and intrinsic folding measures. *Bioinformatics*, **23**(11), 1321–30.
- Rukshan, B. and Vasile, P. (2009). micropred: effective classification of pre-miRNAs for human miRNA gene prediction. *Bioinformatics*, **25**(8), 989–995.
- Stegmayer, G., Pividori, M., and Milone, D. H. (2015). A very simple and fast way to access and validate algorithms in reproducible research. *Briefings in Bioinformatics*. Advance Access published July 28, 2015, doi: 10.1093/bib/bbv054.
- Terai, G., Komori, T., Asai, K., and Kin, T. (2007). mirrim: a novel system to find conserved mirnas with high sensitivity and specificity. *Rna*, **13**(12), 2081–2090.
- Xuan, P., Guo, M., Wang, J., Wang, C., Liu, X., and Liu, Y. (2011). Genetic algorithm-based efficient feature selection for classification of pre-miRNAs. *Genet. Mol. Res.*, **10** (2), 588–603.
- Xue, C., Li, F., He, T., Liu, G., Li, Y., and Zhang, X. (2005). Classification of real and pseudo microRNA precursors using local structure-sequence features and support vector machine. *BMC Bioinformatics*, **6**(1), 310.
- Yousef, M., Nebozhyn, M., Shatkay, H., Kanterakis, S., Showe, L., and Showe, M. (2006). Combining multi-species genomic data for microRNA identification using a naive bayes classifier. *Bioinformatics*, **22** (11), 1325–1334.

Zhang, X., Song, X., and Wang, H. (2010). Characteristic comparison between two types of mirna precursors in metazoan species. *Biosystems*, **100**(2), 144–149.

Zuker, M. and Stiegler, P. (1981). Optimal computer folding of large rna sequences using thermodynamic and auxiliary information. *Nucl Acid*, **9**, 133–148.

miRNAfe detailed feature list

Cristian A. Yones, Georgina Stegnayer, Laura Kamenetzky, and Diego H. Milone

1 Sequence

Feature name	Abbreviation	Brief description	Reference	miRNAfe function name	Vector length
1. Length of sequence	l		[3] ^(ap) ¹	sequence_length	1
2. Nucleotide proportion	$A\%$, $C\%$, $G\%$, $U\%$	Ratio of each base in the sequence	[11] ^(a)	nt_proportion	4
3. Dinucleotide ratio	$AA\%$, $AU\%$, \dots , $GC\%$, $GG\%$	Ratio of dinucleotide elements of each kind.	[10] ^(a) , [14] ^(a)	dinucleotide_proportion	16
4. G+C content	-	Aggregated proportion of guanine and cytosine on the sequence $G + C_{content} = \frac{G + C}{G + C + A + U}$	[4] ^(a) , [10] ^(a) , [3] ^(ap) , [14] ^(a)	gc_content	1
5. G/C ratio	-	Ratio of guanine over cytosine $G/C_{ratio} = \frac{G}{C}$	[6] ^(a)	gc_ratio	1

¹the features of this reference were used in: (a) animals, (p) plants and/or (v) viruses.

2 Secondary structure

Feature name	Abbreviation	Brief description	Reference	miRNAfe function name	Vector length
1. Triplets	-	Vector of 32 elements with the triplets frequency. A triplet is an element formed with the structure composition (paired or not paired) of three adjacent nucleotides and the base of the middle. An example of these elements is ‘((A’, where the parenthesis represent a paired nucleotide, a dot a not paired one and the letter is the base of the middle nucleotide	[15] ^(a) , [9] ^(ap) , [3] ^(ap) , [6] ^(a)	triplets	32
2. Huang elements proportion	-	This feature uses Huang's notation. It is a vector with 10 elements where each one is the proportion of a Huang element (“—”, “==”, “:=”, “^”, “_”, “^=”, “_=”, “::”, “::=”, “::^” and “::^=”).	[5] ^(a)	huang_elements_proportion	10
3. Huang's pMatch ratio	pMatch	This feature use Huang's notation. <i>pMatch</i> indicates the base pairing and is calculated over putative mature miRNA, selected as the 22 nucleotide region where it is maximum.	[5] ^(a)	huang_ratios	1
4. Huang's pMismatch ratio	pMismatch	This feature use Huang's notation and is calculated over putative mature miRNA, selected as the 22 nucleotide region where <i>pMismatch</i> is maximum. <i>pMismatch</i> represents the frequency of non-pairing base pairs (indicated by the size of the interior loops).	[5] ^(a)	huang_ratios	1

Feature name	Abbreviation	Brief description	Reference	miRNAfe function name	Vector length
5. Huang's <i>pDI</i> ratio	<i>pDI</i>	This feature use Huang's notation and is calculated over putative mature miRNA, selected as the 22 nucleotide region where <i>pIMatch</i> is maximum. <i>pDI</i> represents the deletion and insertion frequencies.	[5] ^(a)	huang_ratios	1
6. Huang's <i>pBulge</i> ratio	<i>pBulge</i>	This feature use Huang's notation and is calculated over putative mature miRNA, selected as the 22 nucleotide region where <i>pIMatch</i> is maximum. <i>pBulge</i> indicates the symmetry of the bulged loops.	[5] ^(a)	huang_ratios	1
7. Steam number	<i>l_s</i>	Number of stems in the secondary structure.	[4] ^(a) , [13] ^(a) , [3] ^(ap)	stem_number	1
8. <i>A – U</i> base pair proportion per stem	$A - U / N_{stems}$	Number of adenine-uracil base pair divided by the number of stems.	[11] ^(a) , [10] ^(a) , [6] ^(a) , [14] ^(a)	bp_proportion_stem	1
9. <i>G – C</i> base pair proportion per stem	$G - C / N_{stems}$	Number of guanine-cytosine base pair divided by the number of stems.	[11] ^(a) , [10] ^(a) , [6] ^(a) , [14] ^(a)	bp_proportion_stem	1
10. <i>G – U</i> base pair proportion per stem	$G - U / N_{stems}$	Number of guanine-uracil base pair divided by the number of stems.	[11] ^(a) , [10] ^(a) , [6] ^(a) , [14] ^(a)	bp_proportion_stem	1

Feature name	Abbreviation	Brief description	Reference	miRNAfe function name	Vector length
11. Average base pair per stem	Avg_BP_Stem	Average of nucleotides per stem.	[11] ^(a) , [10] ^(a) , [6] ^(a) , [14] ^(a)	avg_bp_stem	1
12. Length of the longest stem	-	Longest region where the pairing is perfect.	[11] ^(a)	longest_stem_length	1
13. Steam region length	l_s	Number of nucleotides in the stem region of the secondary structure.	[4] ^(a) , [13] ^(a) , [3] ^(ap)	stem_length	1
14. Terminal loop length	l_h	Amount of nucleotides not paired in the terminal loop of the secondary structure $l_h = l - l_s.$	[4] ^(a) , [13] ^(a) , [3] ^(ap)	terminal_loop_length	1
15. Bulges number	N_b		[16] ^(apv)	bulge_number	1
16. Loop number	N_l	Total number of loops, including the terminal loop.	[16] ^(apv) , [3] ^(ap)	loops_number	1
17. Longest loop length	l_{ll}		[3] ^(ap)	longest_loop_length	1
18. Asymmetric loops number	N_{al}		[16] ^(apv)	aloops_number	1
19. Symmetric loops number	N_{sl}		[3] ^(ap)	sloops_number	1
20. Nucleotides in symmetric loops	N_{nsl}		[11] ^(a) , [3] ^(ap)	nt_sloops	1

Feature name	Abbreviation	Brief description	Reference	miRNAsafe function name	Vector length
21. Nucleotides in asymmetric loops	N_{nsl}		[11] ^(a)	<code>nt_aloops</code>	1
22. Longest symmetric region	-	Length and distance to terminal loop of the symmetric region without asymmetric loops or bulges. The symmetric loops are allowed.	[11] ^(a)	<code>longest_simmetric_region</code>	1
23. Average length of symmetric loops	-		[11] ^(a)	<code>avg_length_sloops</code>	1
24. Average length of asymmetric loops	-		[11] ^(a)	<code>avg_length_aloops</code>	1
25. Number of bulges of length 1 to 7 and >7	-	Vector with the number of bulges of length 1, 2, ..., 7 and greater than 7.	[16] ^(apv)	<code>nbulge_length</code>	8
26. Number of loops of length 1 to 7 and >7	-	Vector with the number of loops of length 1, 2, ..., 7 and greater than 7.	[16] ^(apv)	<code>nloops_length</code>	8
27. Base pair number	nP	Number of base pair, i.e. number of paired nucleotides divided by 2	[16] ^(apv)	<code>bp_number</code>	1
28. Adjusted base pair proportion	dP	Number of base pair divided by the nucleotide number.	[8], [10] ^(a) , [6] ^(a) , [14] ^(a)	<code>dP</code>	1
29. A – U pair proportion	$A - U\%$	Proportion of adenine-uracil over the total number of base pairs.	[11] ^(a) , [10] ^(a) , [6] ^(a) , [14] ^(a)	<code>bp_proportion</code>	1

Feature name	Abbreviation	Brief description	Reference	miRNAfe function name	Vector length
30. $G - C$ pair proportion	$G - C\%$	Proportion of guanine-cytosine over the total number of base pairs.	[11] ^(a) , [10] ^(a) , [6] ^(a) , [14] ^(a)	bp_proportion	1
31. $G - U$ pair proportion	$G - U\%$	Proportion of guanine-uracil over the total number of base pairs.	[11] ^(a) , [10] ^(a) , [6] ^(a) , [14] ^(a)	bp_proportion	1
32. $G + C$ content in the terminal loop	-	Aggregated proportion of guanine and cytosine on the terminal loop.	[3] ^(ap)	gc_content_loop	1
33. Reads count	-	The number of reads that match with the stem region of the analyzed sequence.	[3] ^(ap)	read_count	1

3 Thermodynamic stability

Feature name	Abbreviation	Brief description	Reference	miRNAfe function name	Vector length
1. Minimum free energy	<i>MFE</i>	Minimum free energy obtained with the algorithm from Zuker, M. y P. Stiegler, 1981.	[1] ^(a) , [9] ^(ap) , [5] ^(a) , [3] ^(ap)	mfe	1
2. Ensemble free energy	<i>EFE</i>	Ensemble free energy obtained with the algorithm from McCaskill, 1990.	[10] ^(a) , [6] ^(a) , [14] ^(a)	eфе	1
3. MFE index 1	<i>MFEI</i>	Ratio between the minimum free energy and the G+C content.	[17], [8] ^(ap) , [10] ^(a) , [6] ^(a) , [14] ^(a)	mfei1	1
4. Difference of MFE and EFE	<i>Diff</i>	Difference between these two values, divided by the sequence length, $Diff = \frac{MFE - EFE}{l}$.	[10] ^(a) , [6] ^(a) , [14] ^(a)	mfe_eфе_difference	1
5. adjusted MFE	<i>dG</i>	Minimum free energy divided by the sequence length.	[2] ^(a) , [8] ^(ap) , [6] ^(a) , [14] ^(a)	dG	1
6. MFE index 2	<i>MFEI₂</i>	Ratio between the dG and the number of stems.	[8] ^(ap) , [10] ^(a) , [6] ^(a) , [14] ^(a)	mfei2	1
7. MFE index 3	<i>MFEI₃</i>	Ratio between the dG and number of loops.	[10] ^(a) , [6] ^(a) , [14] ^(a)	mfei3	1
8. MFE index 4	<i>MFEI₄</i>	Ratio between the dG and the G+C content.	[10] ^(a) , [6] ^(a) , [14] ^(a)	mfei4	1

Feature name	Abbreviation	Brief description	Reference	miRNAfe function name	Vector length
9. Adjusted Shannon's entropy	dQ	Characterize the probability of base pairing in a secondary structure as a chaotic dynamic system	[8] ^(ap) , [10] ^(a) , [14] ^(a)	dQ	1
		$dQ = \frac{1}{l} \sum_{i < j} p_{ij} \log_2 p_{ij},$			
		where p_{ij} is the probability of pairing of nucleotides i and j . This value is calculated with the algorithm from McCaskill, 1990. Low values of dQ correspond to distributions dominated by a few bases likely to be matched. These bases are better predicted than those that have multiple alternative states.			
10. Adjusted base pair distance	dD	It is the base pair distance for all pairs of structures inferred from the sequence	[8] ^(ap) , [10] ^(a) , [14] ^(a)	dD	1
		$dd = \frac{1}{l} \sum_{i < j} p_{ij} (1 - p_{ij}),$			
11. Ensemble frequency in the set	Freq	Obtained with the algorithm from McCaskill, 1990.	[10] ^(a) , [6] ^(a) , [14] ^(a)	ensemble_frequency	1
12. Set diversity	Diversity	Obtained with the algorithm from McCaskill, 1990.	[10] ^(a) , [6] ^(a) , [14] ^(a)	diversity	1
13. Stem 5' potential	P^L	It is the maximum probability of pairing a nucleotide with other that is on the 5' direction.	[2] ^(a)	stem5_potential	Variable
		$P^L_i = \max_{j < i} p_{ij},$			
		where p_{ij} is the same defined for dQ.			

Feature name	Abbreviation	Brief description	Reference	miRNAs function name	Vector length
14. Stem 3' potential	P^R	It is the maximum probability of pairing a nucleotide with other than the corresponding in the 3' direction.	[2] ^(a)	<code>stem5_potential</code>	Variable
		$Pl_i = \max_{j>i} p_{ij},$ where p_{ij} is the same defined for dQ.			
15. Loop potential	V'	It is a vector where each element measures how likely a nucleotide can be part of the terminal loop	[2] ^(a)	<code>loop_potential</code>	Variable
		$V'_i = \sum_j \omega_{i-j} \left[\sum_k p_{j-k, j+k} + p_{j-k+1, j+k} \right]$ where p_{ij} is the same defined for dQ and ω is a smoothing window.			

4 Statistical stability

Feature name	Abbreviation	Brief description	Reference	miRNAfe function name	Vector length
1. Standard score of the MFE	$zMFE$	Minimum free energy normalized with z-score.	[4] ^(a) , [2] ^(a)	zMFE	1
2. Standard score of the FFE	$zFFE$	Ensemble free energy normalized with z-score.	[6] ^(a)	zFFE	1
3. Standard score of the dG	zG	Adjusted minimum free energy normalized with z-score.	[8] ^(ap) , [10] ^(a) , [14] ^(a)	zG	1
4. Standard score of the Shannon's entropy	zQ	Adjusted Shannon's entropy normalized with z-score.	[8] ^(ap) , [10] ^(a) , [14] ^(a)	zQ	1
5. Standard score of the base pair proportion	zP	Base pair proportion adjusted and normalized using z-score.	[8] ^(ap) , [10] ^(a) , [14] ^(a)	zP	1
6. Standard score of the base pair distance	zD	Adjusted base pair distance normalized using z-score.	[6] ^(a)	zD	1
7. Monte Carlo and randomization test over MFE	$pMFE$	p-value of the ensemble free energy.	[1] ^(ap)	pMFE	1
8. Monte Carlo and randomization test over FFE	$pFFE$	p-value of the minimum free energy.	[6] ^(a)	pFFE	1

5 Phylogenetic conservation

Feature name	Abbreviation	Brief description	Reference	miRNAfe function name	Vector length
1. Mutation frequency	—	Number of mutation (differences) between two sequences of RNA. Only applicable to alignments of two sequences.	[5] ^(a)	mutation_frequency	1
2. Column entropy of the 5' arm	$S_{5'}$	Shannon's entropy of the 5' arm	[4] ^(a)	column_entropy	1
3. Column entropy of the 3' arm	$S_{3'}$	Shannon's entropy of the 3' arm	[4] ^(a)	column_entropy	1
4. Column entropy of the loop region	S_0	Shannon's entropy of the terminal loop	[4] ^(a)	column_entropy	1
5. Minimum entropy	S_{min}	Minimum entropy calculated over a region of 21 nucleotides.	[4] ^(a)	column_entropy	1
6. Secondary structure differences	V_{strc}	Difference between the secondary structures of two sequences caused by mutations divided by the number of differences between sequences	[5] ^(a)	se_difference	1
7. Average minimum free energy	\bar{E}	Mean of the minimum free energies of the sequences that are part of the alignment.	[4] ^(a)	mean_mfe	1
8. MFE difference	$VMFE$	Difference between the minimum free energy of two aligned sequences divided by the number of differences between the sequences.	[5] ^(a)	mfe_difference	1

Feature name	Abbreviation	Brief description	Reference	miRNAfe function name	Vector length
9. Average of dG	\bar{e}	Mean of the adjusted minimum free energies of aligned sequences.	[4] ^(a)	mean_dG	1
10. Average of $MFEI_1$	$\bar{\eta}$	Mean of the MFE_1 of the aligned sequences.	[4] ^(a)	mean_mfei1	1
11. Free energy of the consensus secondary structure	E_{cons}		[4] ^(a) , [7] ^(a)	mfe_consensus	1
12. Conservation of the 3' arm	—	Number of bases conserved in two or more sequences in the 3' arm, without the 10 first bases of the substring.	[4] ^(a) , [7] ^(a)	conservation_3	1
13. Conservation of the 5' arm	—	Number of bases conserved in two or more sequences in the 5' arm, without the 10 first bases of the substring.	[7] ^(a)	conservation_5	1
14. Conservation score	CS	Conservation score of the alignment of sequences. Internally uses the software PhyloFit ² . This score is calculated using two Markov processes, one that moves in the time dimension (over the branches of the evolution tree), and the other in space dimension (over the sequence).	[2] ^(a) [12]	conservation_score	1

²<http://compgen.bscb.cornell.edu/phast/index.php>

6 22-nt substring analysis

Feature name	Abbreviation	Brief description	Reference	miRNAfe function name	Vector length
1. Base pair probability	—	Sum of base-pairing probability over the substring.	[7] ^(a)	ss_base_pair	Variable
2. Not paired bases	—	Sum of not paired bases on the substring.	[3] ^(ap)	ss_not-paired	Variable
3. Extension base pair probability	—	Sum of base-pairing probability on the secondary structure, without probabilities of the nucleotides on the substring.	[7] ^(a)	ss_extension_base_pair	Variable
4. Bulge symmetry	—	The difference between the amount of not paired bases on each arm of the substring.	[7] ^(a)	ss_bulge_symmetry	Variable
5. Terminal loop distance	—	Distance from the substring to the terminal loop.	[7] ^(a) , [13] ^(a)	ss_loop-distance	Variable

miRNAfe validation of feature extraction processes

Cristian A. Yones, Georgina Stegmayer, Laura Kamenetzky, and Diego H. Milone

In the next table, the software used for comparisons and their corresponding references are presented.

Feature	Software used
Triplets	MiRFinder [5]
Huang ratios	MiRFinder [5]
Huang elements proportion	MiRFinder [5]
$G + C_{content}$	microPred [10]
Dinucleotide proportion	microPred [10]
$MFEI_1$	genRNAsStats and RNAspectral of miPred [6]
$MFEI_2$	genRNAsStats and RNAspectral of miPred [6]
$MFEI_3$	microPred [10]
$MFEI_4$	genRNAsStats and RNAspectral of miPred [6]
MFE difference	MiRFinder [5]
Secondary structure difference	MiRFinder [5]
Mutation frequency	MiRFinder [5]
$zMFE$	genRandomRNA of miPred [6]
$zEFE$	genRandomRNA of miPred [6]
zQ	genRandomRNA of miPred [6]
zP	genRandomRNA of miPred [6]
zG	genRandomRNA of miPred [6]
zD	genRandomRNA of miPred [6]
$pEFE$	genRandomRNA of miPred [6]
$pMFE$	genRandomRNA of miPred [6]

References

- [1] Bonnet, E, J Wuyts, P Rouzé, and Y Van de Peer: *Evidence that microRNA precursors, unlike other non-coding RNAs, have lower folding free energies than random sequences.* Bioinformatics, 20 (17):2911–2917, 2004.
- [2] Goro, T, K Takashi, A Kiyoshi, and K Taishin: *mirrnm: A novel system to find conserved miRNAs with high sensitivity and specificity.* RNA, 13 (12):2081–2090, 2007.
- [3] Hackenberg, M, M Sturm, and D Langenberger: *miRanalyzer: a microRNA detection and analysis tool for next-generation sequencing experiments.* Nucleic Acids Research, 37:68–76, 2009.
- [4] Hertel, J and PF Stadler: *Hairpins in a haystack: recognizing microRNA precursors in comparative genomics data.* Bioinformatics, 22 (14):e197–e202, 2006.
- [5] Huang, TH, B Fan, M Rothschild, ZL Hu, K Li, and SH Zhao: *MirRFinder: an improved approach and software implementation for genome-wide fast microRNA precursor scans.* BMC Bioinformatics, 8(1):341, 2007.
- [6] Jiandong, D, Z Shuigeng, and G Jihong: *MirnSVM: towards better prediction of microRNA precursors using an ensemble SVM classifier with multi-loop features.* BMC Bioinformatics, 11 (11):11, 2010.
- [7] Lim, LP, NC Lau, EG Weinstein, A Abdelhakim, S Yekta, MW Rhoades, CB Burge, and DP Bartel: *The microRNAs of caenorhabditis elegans.* Genes & development, 17(8):991–1008, 2003.
- [8] Ng, KLS and SK Mishra: *De novo SVM classification of precursor microRNAs from genomic pseudo hairpins using global and intrinsic folding measures.* Bioinformatics, 23(11):1321–30, 2007.
- [9] Peng, J, W Haonan, W Wenkai, M Wei, S Xiao, and L Zuhong: *MiPred: classification of real and pseudo microRNA precursors using random forest prediction model with combined features.* Nucleic Acids Research, 35:339–44, 2007.
- [10] Rulkhan, B and P Vasile: *micropred: effective classification of pre-miRNAs for human miRNA gene prediction.* Bioinformatics, 25(8):989–995, 2009.
- [11] Sewer, A, N Paul, P Landgraf, A Aravin, S Pfeffer, MJ Brownstein, T Tuschl, E van Niniwegen, and M Zavolan: *Identification of clustered microRNAs using an ab initio prediction method.* BMC Bioinformatics, 6:267, 2005.
- [12] Siepel, A and D Haussler: *Phylogenetic hidden Markov models.* In *In statistical methods in molecular evolution*, pages 325–351. Springer, 2005.
- [13] Snorre, AH, S Ola, and S Pal: *Reliable prediction of drosha processing sites improves microRNA gene prediction.* Bioinformatics, 23(2):142–149, 2007.
- [14] Xuan, P, MZ Guo, J Wang, CY Wang, XY Liu, and Y Liu: *Genetic algorithm-based efficient feature selection for classification of pre-miRNAs.* Genet. Mol. Res., 10 (2):588–603, 2011.
- [15] Xue, C, F Li, T He, GP Liu, Y Li, and X Zhang: *Classification of real and pseudo microRNA precursors using local structure-sequence features and support vector machine.* BMC Bioinformatics, 6(1):310, 2005.
- [16] Yousef, M, M Nebozhyn, H Shatkaty, S Kanterakis, LC Showe, and MK Showe: *Combining multi-species genomic data for microRNA identification using a naive bayes classifier.* Bioinformatics, 22 (11):1325–1334, 2006.
- [17] Zhang, BH, XP Pan, SB Cox, GP Cobb, and TA Anderson: *Evidence that miRNAs are different from other RNAs.* Cell. Mol. Life Sci., 63(2):46–254, 2006.

Genome-wide pre-miRNA discovery from few labeled examples

Genome-wide pre-miRNA discovery from few labeled examples

C. Yones, G. Stegmayer and D. H. Milone

Research Institute for Signals, Systems and Computational Intelligence, sinc(*i*), FICH-UNL, CONICET,
Santa Fe, Argentina.

Abstract

Motivation: Although many machine learning techniques have been proposed for distinguishing miRNA hairpins from other stem-loop sequences, most of the current methods use supervised learning, which requires a very good set of positive and negative examples. Those methods have important practical limitations when they have to be applied to a real prediction task. First, there is the challenge of dealing with a scarce number of positive (well-known) pre-miRNA examples. Secondly, it is very difficult to build a good set of negative examples for representing the full spectrum of non-miRNA sequences. Thirdly, in any genome, there is a huge class imbalance (1:10000) that is well-known for particularly affecting supervised classifiers.

Results: To enable efficient and speedy genome-wide predictions of novel miRNAs, we present MiRNAss, which is a novel method based on semi-supervised learning. It takes advantage of the information provided by the unlabeled stem-loops, thereby improving the prediction rates, even when the number of labeled examples is low and not representative of the classes. An automatic method for searching negative examples to initialize the algorithm is also proposed so as to spare the user this difficult task. MiRNAss obtained better prediction rates and shorter execution times than state-of-the-art supervised methods. It was validated with genome-wide data from three model species, with more than one million of hairpin sequences each, thereby demonstrating its applicability to a real prediction task.

Availability: Can be downloaded from <https://CRAN.R-project.org/package=miRNAss>. In addition, a web-demo is available at <http://fich.unl.edu.ar/sinc/web-demo/mirnass>. All the datasets that were used in this study and the sets of predicted pre-miRNA are available on <http://sourceforge.net/projects/sourcesinc/files/mirnass>

Contact: cyones@sinc.unl.edu.ar.

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Over the last decade, a number of experimental and computational approaches were used to identify novel miRNAs. Experimental methods such as direct cloning are capable of detecting only abundant miRNAs (Kleftogiannis *et al.*, 2013). Moreover, those expressed exclusively in certain stages of an organism, or in specific tissues, usually remain undetected. Most widely used approaches are hybrid computational/experimental methods, which take also advantage of the available deep sequencing

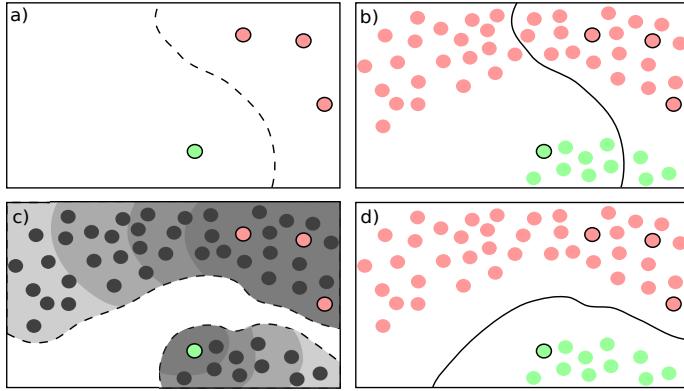


Figure 1: Prediction example: supervised (a-b) versus semi-supervised (c-d) approaches.

data (An *et al.*, 2013). The computational approaches used for pre-miRNA identification may be divided into two groups: homology search methods and machine learning methods. The first ones rely on the conservation of miRNAs between closely related species and thus cannot be used for miRNAs that are specific to one species (Ng and Mishra, 2007). Machine learning methods use a different approach, which involves first extracting features from the well-known pre-miRNAs, the sequences that are used as negative examples, and the unknown/unlabeled sequences to be classified. The features that are used to represent RNA sequences were widely studied (Lopes *et al.*, 2014; Yones *et al.*, 2015). After feature extraction, many types of different classifiers were applied to this problem: support vector machines, random forest, classifier ensembles, and hidden Markov models, among others (Kleftogiannis *et al.*, 2013). The existing methods based on this supervised approach of machine learning have many drawbacks. In the first place, although the number of non-miRNA sequences that can be found in any genome is large, a recent study (Wei *et al.*, 2014) analyzed the importance of representative negative samples in miRNA prediction and how this limits the performance of state-of-the-art methods. In most cases, the negative sets that are used to fit classification models are defined mainly with sequences taken from protein coding regions, mRNAs, or other regions where miRNAs are not expected to be found (Peace *et al.*, 2015). Nothing can ensure that these sequences are good representatives of all possible non-miRNA sequences, or that they are close enough to the boundaries of the true miRNA class. For example, if the prediction power of a classifier is tested on a dataset having miRNAs and ncRNAs, what is actually being measured is the capability to differentiate miRNAs from ncRNAs, but the classifier may still fail to correctly discard other non-miRNA sequences. To the best of our knowledge, no current state-of-the-art method has reported error rates considering this crucial point. Moreover, if sequences from these well-known sources of negative examples could be discarded a priori with a set of simple rules, training a model to differentiate between these two groups of sequences would be useless. To address this issue, some methods used sequences taken from random positions of a genome as negative examples (Wenyuan *et al.*, 2013; Gudyś *et al.*, 2013). However, in this case, nothing can ensure that any really novel miRNA would not be wrongly taken as a negative example. In both negative set construction strategies, the negative examples are not representative of the whole negative class; and, what is even worse, some of them might actually be false negatives.

Another issue is that the existing tools are generally built using a classifier trained with a model

species (Batuwita and Palade, 2009) or a mixture of sequences from several species (Gudyś *et al.*, 2013); however, the miRNA prediction problem is a species-dependent task (Lopes *et al.*, 2016). Therefore, using pre-trained models cannot ensure acceptable results when predicting novel pre-miRNAs in other species that are different from the ones that were used for training. Although some tools can be re-trained, this presents some difficulties. As discussed above, negative examples are hard to define. Besides, the number of well-known positive examples is very low, since a very small proportion of true miRNAs is expected in a genome. This is an important problem in novel and non-model species, where the number of annotated pre-miRNAs might be even lower. Therefore, regardless of how negative examples are obtained or defined, and because of the natural class imbalance in any genome, thousands or even millions of unlabeled sequences are expected to be negative. This poses a high class imbalance problem, which most machine learning algorithms cannot face correctly.

In order to appropriately address these issues, we present a new method named miRNAss, which uses a semi-supervised approach to learn the latent distribution of hairpin features of the genome under analysis. This method takes advantage of unknown sequences to improve the prediction rates, even when there are just a few positive examples, and when the negative examples are unreliable or are not good representatives of its class. Furthermore, it can automatically search for negative examples if the user is unable to provide them. In addition, miRNAss automatically optimizes the threshold that defines the class boundaries and thus can separate the pre-miRNAs from other groups of sequences, in spite of the high class imbalance. Finally, we will show that the method can handle large volumes of data.

2 Methods

2.1 Semi-supervised and transductive learning

Semi-supervised learning not only uses labeled training examples but also learns from unlabeled examples to improve the classification performance. As the unlabeled examples provide extra information about the latent distribution of the data, good classification rates can be achieved even when the amount of labeled data is very low compared with unlabeled data, or when it is unrepresentative of its class (Chapelle *et al.*, 2006). Fig. 1 shows a very simple scenario with four labeled examples for training: three negative examples (in red) and one positive example (in green). If a classic supervised method is used here, a decision boundary like the dashed line in Fig. 1.a is obtained. This classifier fails over the unknown sequences (Fig. 1.b), because the training sequences are very few and actually unrepresentative of each class. If a semi-supervised scheme is employed instead (Fig. 1.c), it is observed that it has access to both labeled and unlabeled sequences, all together at the same time, where two real groups are hidden beneath the latent distribution of the data. Thus, the semi-supervised predictor can successfully separate the two classes (Fig. 1.d).

The concept of transductive learning is closely related to semi-supervised learning. In this setting, specific training cases are used to directly make inferences on test cases. Unlike most common inductive methods, where the algorithm has two stages (training and testing), transductive methods have only one stage, as shown in Fig. 2. In the inductive scheme (a), the learning algorithm uses training sequences, all labeled, to fit a model. In the second stage, the fitted model is used to make a prediction over unknown sequences. In the transductive scheme (b), both labeled and unlabeled sequences are processed together, in the same stage. As a result, a prediction over the unlabeled sequences is obtained, but no decision function, trained model, or classification rule are

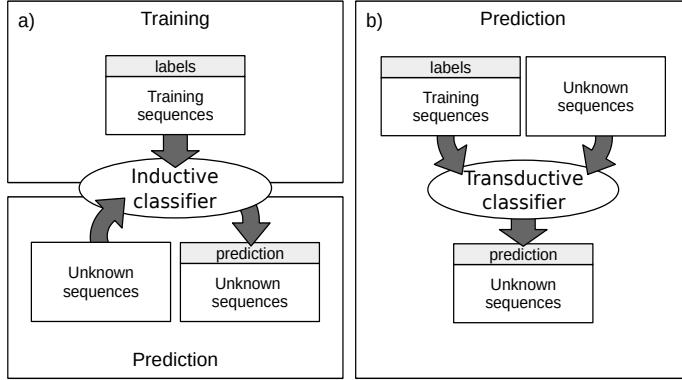


Figure 2: Inductive and transductive learning. a) Traditional inductive scheme with two separate stages; b) Transductive scheme with only one stage.

returned as output. While an inductive method infers a decision function that can be used to predict labels of any sequence, the transduction model directly estimates the set of labels for the unknown sequences (Chapelle *et al.*, 2006). Nevertheless, a transductive classifier can be analyzed after training similarly to an inductive one. For example, in a graph based method, the adjacencies among the samples labeled in the transductive stage could be used for knowledge extraction (Enright and Ouzounis, 2001; Novák *et al.*, 2010). The semi-supervised method proposed in this work uses such transductive scheme. Therefore, training examples and unknown sequences are processed together, every time a prediction has to be made. This is not a problem in pre-miRNA prediction, because it is preferable for the model to be fed with all the sequences from the species under study (or a closely related one) in order to learn its specific characteristics (Lopes *et al.*, 2016). It should be noted that the validation of any transductive method is quite different from a standard (inductive) one. Since training and prediction are combined in one stage, training sequences are provided with labels, and test data are also given but *without* labels. This special kind of validation is used in all semi-supervised algorithms (Chapelle *et al.*, 2006), where the labels of test cases are never used in any step of the prediction stage.

2.2 The miRNAss method

Over the last decade, one of the most active areas in semi-supervised learning has been based on graphs, where each node represents a data point, and an edge connects nodes if they are in some way similar. Then, using the labeled nodes, predicted labels are obtained for the rest of the unlabeled nodes. These methods have shown good prediction rates (Joachims *et al.*, 2003) and have been able to handle large volumes of data, because they can be implemented using a sparse matrix in $\mathcal{O}(n)$ memory, with n number of analyzed sequences. The most computationally expensive part is the graph construction; however, this can be done by brute force in $\mathcal{O}(n^2 \log n)$, calculating the distances between all sequences and searching for the smallest k for each one in $\mathcal{O}(\log n)$. MiRNAss receives as input a set of m -dimensional feature vectors \mathbf{x}_i , which represent sequences, and a corresponding vector of labels ℓ . The i -th element in the vector of labels has a positive value γ_+ if the i -th sequence is a well-known pre-miRNA, a negative value γ_- if it is not a pre-miRNA, and zero if it is an unknown sequence that has to be classified (predicted) by the method. The method has

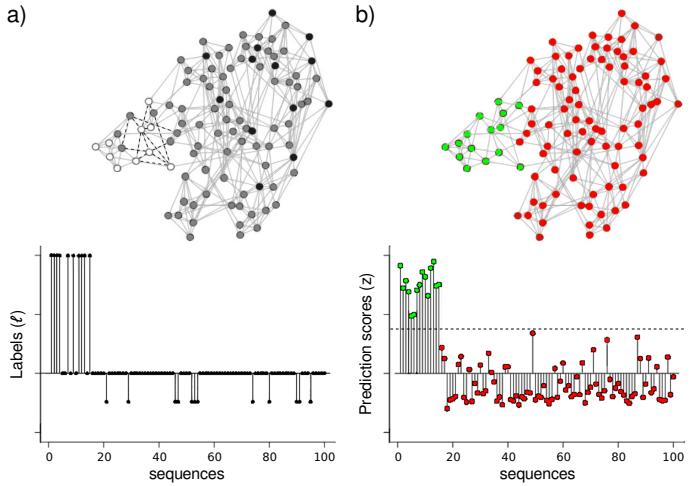


Figure 3: Evolution of the graph, the label vector, and the vector of prediction scores in the two steps of miRNAss: a) Graph construction and negative set initialization; b) Estimation of prediction scores and thresholding.

four steps: 1) construction of a graph where each node represents a sequence; 2) search for negative examples, if they are not provided; 3) estimation of the prediction scores for each node in the graph; and 4) estimation of an optimal threshold to finally separate the sequences into two classes. Each step is explained in detail in the following subsections.

Fig. 3 shows an example of the evolution of the method. In this example, 15 sequences are true pre-miRNAs and the rest are non-miRNA sequences. In Fig. 3.a, the graph is constructed and ℓ has positive values (γ_+), representing the well-known pre-miRNAs examples. In addition, some nodes that are topologically far from the positive examples are labeled as negative examples (γ_-). The nodes of the graph are colored according to the values in the ℓ vector. In Fig. 3.b, prediction scores are estimated for all the sequences, taking into account that: i) they have to be smooth; that is, topologically close sequences in the graph must have similar prediction scores; and ii) the scores have to be similar to the non-zero values given in the label vector ℓ . Finally, using the prediction scores assigned to the labeled examples, an optimal threshold is estimated in order to separate the pre-miRNAs from the other sequences. The sequences that pass the threshold are colored in green; the rest, in red.

2.2.1 Graph construction

To build the graph, each node represents a sequence, and an edge joins two nodes if the sequences can be considered similar. The euclidean distance between the feature vectors is used to make this measurement. If a feature does not really help to discriminate between classes, it may worsen the performance of the classifier. Therefore, it is important to pre-process the feature vectors to properly weight each feature according to its prediction power. An algorithm that has shown to improve results in classifiers that are sensitive to distance function is the RELIEF-F algorithm (Kononenko, 1994; Wettschereck *et al.*, 1997). Furthermore, it is computationally efficient and can be used for large volumes of data. It works as follows: starting with a vector of weights with m

zeros, for each example it searches for the closest same-class example and the closest different-class example. Then, it increases the weights of features that are similar to the same-class example and different to the different-class example. Conversely, the weights of features that are different to the same-class example or similar to the different-class example are decreased. The result is called relevance vector and has high values for the most discriminative features. If the relevance of a feature is negative, it does not help to discriminate between classes; therefore can be removed. The rest of the features are scaled by their relevance score in order to give more weight to the most discriminative ones.

For graph construction, a common choice is the k-nearest neighbors (KNN) algorithm, because it is simple, fast, memory-efficient, and easily parallelizable. KNN constructs a similarity-weighted matrix

$$a_{ij} = \begin{cases} \frac{\mu}{\mu + ||\mathbf{x}_i - \mathbf{x}_j||^2} & \text{if } \mathbf{x}_j \in \mathcal{K}(\mathbf{x}_i) \text{ and } \ell_i \ell_j \geq 0 \\ 0 & \text{in other cases,} \end{cases} \quad (1)$$

where \mathbf{x}_i is the feature vector corresponding to the i -th sequence, $\mathcal{K}(\mathbf{x}_i)$ is the set of the k-nearest neighbors of \mathbf{x}_i , and μ is the mean of the distances between the connected sequences used to normalize the edge weights.

2.2.2 Automatic search for negative examples

If only positive examples are provided, the graph adjacency matrix can be used to label a set of sequences as negative. The key idea is to randomly select some unlabeled sequences among the most dissimilar from the positive examples. That way, the probability of incorrectly labeling a well-known pre-miRNA as a negative example will be low. For that purpose, a measure of the similarity of each sequence to the positive examples is calculated using the topological distance. This similarity vector, called \mathbf{s} , is defined +1 in the elements corresponding to positive examples. The unlabeled nodes are initialized to 0. Then, an iterative method can be used to update the similarities in \mathbf{s} (see Algorithm 1). In each iteration and for every node, the corresponding similarity of each neighbor is multiplied by the corresponding edge weights that connect them. The maximum value from the results obtained for each neighbors is then compared with the current similarity value of the node. If it is higher, the similarity value of the node is updated. When there are no more changes in \mathbf{s} , T sequences are sampled using $p_i = e^{1-s_i} - 1$ and labeled as negative examples.

2.2.3 Estimation of the prediction scores

In the third step, the prediction scores are calculated by solving an optimization problem (Joachims *et al.*, 2003). As stated earlier, two points should be considered: i) the prediction scores must be topologically smooth; and ii) the predictions must be similar to the known labels. To make prediction smooth, the square of the differences between the prediction scores of adjacent sequences is minimized. A convenient representation for easily calculating these differences is the normalized Laplacian graph (Shi and Malik, 2000), defined as $L = I - D^{-1/2}AD^{-1/2}$, where D is the degree matrix defined as $d_{ij} = \sum_{k=0}^n a_{ik}$ if $i = j$, and zero in other cases. The Laplacian graph has a useful property for measuring the smoothness of the solution. Suppose $\mathbf{z} \in \mathbb{R}^N$, with one prediction for

Algorithm 1 Automatic search for negative examples

Input: adjacency matrix A , label vector ℓ with zeros and positive values only and the number of negative examples to label T .

Output: the vector ℓ with T negative labels assigned.

- 1: $s_i = \begin{cases} 1 & \text{if } \ell_i > 0 \\ 0 & \text{in other cases} \end{cases}$
 - 2: **repeat**
 - 3: $s_i = \max_{\forall j \neq i} \{s_i, a_{ij}s_j\}, \quad \forall i$
 - 4: **until** there are no changes in s
 - 5: $p_i = e^{1-s_i} - 1, \quad \forall i$
 - 6: Sample T elements of ℓ using \mathbf{p} as selection weights
 - 7: Label selected elements as negative class
 - 8: **return** ℓ
-

each node of the graph. Then,

$$\begin{aligned}
 \mathbf{z}^T L \mathbf{z} &= \mathbf{z}^T I \mathbf{z} - \mathbf{z}^T D^{-1/2} A D^{-1/2} \mathbf{z} = \\
 &= \sum_i^n z_i^2 - \sum_i^n \sum_j^n \frac{z_i}{\sqrt{d_{ii}}} \frac{z_j}{\sqrt{d_{jj}}} a_{ij} = \\
 &= \frac{1}{2} \sum_i^n \sum_j^n a_{ij} \left(\frac{z_i}{\sqrt{d_{ii}}} - \frac{z_j}{\sqrt{d_{jj}}} \right)^2.
 \end{aligned} \tag{2}$$

This last expression shows that $\mathbf{z}^T L \mathbf{z}$ measures the squared difference between predictions z_i and z_j , weighted by a_{ij} . If sequences i and j are similar, and thus a_{ij} has a relative high value, any difference between the two predictions will have a high cost. If there is no edge connecting the two sequences, $a_{ij} = 0$ and the difference between predictions is ignored. Furthermore, it should be noted that predictions are weighted by the inverse of the square root of the node degree. As a result, nodes with a small degree are considered as important as highly connected nodes.

To minimize the inconsistency between predictions and well-known labels, the squared difference between predictions and non-zero labels in ℓ is also required in the objective function. Therefore, it has two terms: the first term measures the non-smoothness of the solution using the normalized Laplacian matrix (unsupervised component), and the second term is the squared difference between predictions and non-zero labels in ℓ (supervised component). To take full advantage of the semi-supervised learning, there should not be strong overlap between the classes to be separated. Nevertheless, if this prerequisite is not fulfilled, the method behaves as any other supervised method in the same conditions. If there is not any clear separation, the first term of the objective function will not have any sharp minimum. Therefore, the second term of the equation (the supervised one) will lead the search. The full optimization problem is defined as

$$\begin{aligned}
 \arg \min_{\mathbf{z}} \quad & \mathbf{z}^T L \mathbf{z} + c(\mathbf{z} - \ell)^T C(\mathbf{z} - \ell) \\
 \text{s.t.} \quad & \mathbf{z}^T \mathbf{1} = 0, \mathbf{z}^T \mathbf{z} = n,
 \end{aligned} \tag{3}$$

where the combination of both restrictions avoids trivial solutions. In the first restriction, the sum of elements of \mathbf{z} is required to be zero; that is, the prediction labels need to have both negative and positive values. The second restriction eliminates scaled versions of the solution that, for our purpose, are all equivalent. The values ℓ_i are set to γ_+ , γ_- , or zero, depending on whether the i -th sequence is a positive, a negative, or an unknown example, respectively. As the objective function forces the values of \mathbf{z} to be close to γ_+ or γ_- , these constants have to be defined such that the optimal solution \mathbf{z}^* be able to satisfy both restrictions of the problem. If n_+ and n_- are the true numbers of positive and negative nodes in the solution, defining $\gamma_+ = \sqrt{n_-/n_+}$ and $\gamma_- = -\sqrt{n_+/n_-}$ makes \mathbf{z}^* satisfy both restrictions. This can be observed by replacing \mathbf{z}^* in the restrictions and assuming that this vector has n_+ elements equal to γ_+ and n_- equal to γ_- . The numbers n_+ and n_- are usually unknown, but they can be easily estimated from the training examples. If positive and negative examples are provided for training, miRNAss will estimate n_+/n_- as the proportion of examples given. If there are only positive examples, n_+ is estimated as twice the number of training sequences and $n_- = n - n_+$. This estimation could be improved using domain knowledge, that is, using the expected number of well-known miRNAs for a given species; however, it is not necessary as miRNAss is not sensitive to these parameters (see Figure S1 in Supplementary Material). Therefore, any value between the number of positive training sequences and four times this number can be used without impact in performance. By default, twice the number of positive training sequences is used as an intermediate value. The constant c in the objective function can be used to set the relative weight of the second term compared with the first one. A large value of c gives a higher penalization to the misclassifications, pushing the prediction scores to values similar to the non-zero labels ℓ . Conversely, if a low value of c is used, the misclassifications are less penalized and the first term dominates the objective function, thus producing a smoother solution. Matrix C in the second term is a diagonal matrix that is zero-valued in the elements corresponding to unknown sequences. This way, the unlabeled sequences are ignored in this term. In the non-zero elements (corresponding to the labeled examples), the value assigned allows different misclassification costs per sequence. This weighting can be used, for example, to assign lower values to pre-miRNAs that have not been experimentally validated or that are unreliable negative examples. It can also be used to avoid misclassification of labeled examples. In Section S2 (of the Supplementary Material) it is proved that if $C_{ii} > (nn_+)/(cn_-)$, misclassifying the i -th sequence will have a greater penalization than any penalization in the not supervised term. Then, it cannot be misclassified. As a default value, the non-zero elements of C are set to 1, for both positive and negative examples.

To solve this optimization problem, we first calculate the spectral decomposition of the Laplacian $L = U\Sigma U^T$. Next, we introduce a new parameter vector \mathbf{w} and replace $\mathbf{z} = U\mathbf{w}$. Since the eigenvector corresponding to the lowest eigenvalue is always constant, the first constraint of the optimization problem becomes equivalent to $w_1 = 0$. If we define V as the matrix with all the eigenvectors except the first one, and H as the diagonal matrix with all eigenvalues except the lowest one, we get the following optimization problem

$$\begin{aligned} \arg \min_{\mathbf{w}} \quad & \mathbf{w}^T H \mathbf{w} + c(V\mathbf{w} - \ell)^T C(V\mathbf{w} - \ell) \\ \text{s.t.} \quad & \mathbf{w}^T \mathbf{w} = n. \end{aligned} \tag{4}$$

Defining $Q = H + cV^T CV$ and $b = cV^T C\ell$, this problem can be rewritten as

$$\begin{aligned} \arg \min_{\mathbf{w}} \quad & \mathbf{w}^T Q \mathbf{w} + 2b^T \mathbf{w} + c\ell^T C\ell \\ \text{s.t.} \quad & \mathbf{w}^T \mathbf{w} = n, \end{aligned} \tag{5}$$

where the last term can be dropped since it is constant. Using Lagrange multipliers, the global minimum of this function occurs in $\mathbf{w}^* = (Q - \lambda^* I)^{-1}\mathbf{b}$. Now, using the results of (Gander *et al.*, 1989), λ^* can be calculated as the lowest eigenvalue of

$$M = \begin{pmatrix} Q & -I \\ \frac{\mathbf{b}\mathbf{b}^T}{n} & Q \end{pmatrix}. \tag{6}$$

From this solution, the predicted labels are calculated as $\mathbf{z}^* = V\mathbf{w}^*$.

2.2.4 Thresholding the prediction scores

Given the high imbalances that are present in genome-wide data, increasing the misclassification cost of the positive class is mandatory. While matrix C can be used to assign different misclassification costs in the optimization process, estimating the threshold to be applied in \mathbf{z} may be a more flexible and efficient method for maximizing a given performance measure. Since the prediction scores are expected to rank sequences properly, thresholding the results of the prediction scores is equivalent to classifying with weighted costs (Mease *et al.*, 2007). In addition, depending on the user's needs, different performance measures may be selected for optimization after the prediction scores are calculated.

A common assessment metric that is used in class imbalanced problems is the geometric mean (\bar{G}) of the sensitivity and the specificity $\bar{G} = \sqrt{S^+ S^-}$ (Batuwita and Palade, 2009; Gudyś *et al.*, 2013), where S^+ is the proportion of positive sequences correctly classified as positive (sensitivity), and S^- is the proportion of negative sequences correctly predicted as negative (specificity). This measure has the advantage of giving the same importance to both negative and positive classes, regardless of the number of elements in each class, and it will be used for a fair comparison with previous method. A better metric is the F-measure $F_1 = 2P S^+ / (P + S^+)$, where P is the precision, that is, the proportion of true positives within the sequences classified as positive. When F_1 is used for threshold optimization, it gives a better P at the cost of a lower S^+ . Given the large class imbalances in the pre-miRNA prediction problem, it is important to take into account the number of false positives. Therefore, F_1 is a better measure for this problem. To find a threshold, the prediction scores obtained for labeled examples may be used to estimate the target measure of performance. It should be pointed out that this is only an estimation, since the unlabeled sequences cannot be used. For this reason, between two consecutive labeled examples in \mathbf{z}^* increasingly sorted by the estimated performance measure, there is a constant region. If the number of labeled examples is low, these constant regions can be relatively wide and cannot be neglected. Therefore, the final threshold is set as the midpoint between the highest and the lowest scores in \mathbf{z}^* which maximizes the performance measure (see Supplementary Figure S3). Once a prediction is made, new nodes can be added to the labeled graph using fast algorithms to find the KNN (Malkov *et al.*, 2014). This allows making predictions over new sequences or extract information from raw data by analyzing

Table 1: Time comparison between miRNAss and HuntMi.

Dataset	Number of sequences			Time comparison		
	Total	miRNA	non-miRNA	HuntMi	miRNAss	Speedup
Virus	1,076	237	839	38 s	1 s	38.00
Arabidopsis	28,590	231	28,359	1,819 s	129 s	14.10
Human	82,634	1,406	81,228	9,873 s	462 s	21.37
Plants	117,101	2,172	114,929	21,561 s	714 s	30.19
Animals	225,207	7,053	218,154	65,762 s	1,834 s	35.85

their adjacencies (Chapelle *et al.*, 2006).

3 Results and discussion

This section presents a set of experiments for testing the performance of miRNAss under different conditions. In the first subsection, a group of experiments performed by other authors were reproduced to compare miRNAss with state-of-the-art supervised methods under controlled conditions. The negative sets were artificially defined by the original authors and the proportion of labeled examples was very high. In the second subsection, the percentage of labeled examples was gradually reduced so as to move a step closer to a real pre-miRNA prediction task. Additionally, miRNAss was tested in a one-class scheme, where no negative examples were known in advance. In the last subsection, miRNAss was applied to a real prediction task from genome-wide data, with no negative examples and only a low number of positive examples available. In this case, there was a huge class imbalance and there were more than one million input sequences to process.

3.1 Comparison on full labeled datasets

The experimental conditions in this section are the same than the state-of-the-art methods used for comparisons. The objective measure used for threshold optimization in all methods was set to F_1 . Since datasets were designed for supervised learning, the majority of sequences are artificially labeled for training, and only a 10% percentage is left unlabeled. Therefore, it is important to point out that the main advantage of our semi-supervised method, which is exploiting unlabeled data to improve results, remained unused here. For a fair comparison, in these experiments the classifiers use the same features. This set is composed by 7 features from miPred (Ng and Mishra, 2007), plus 14 from microPred (Batuwita and Palade, 2009) and 7 added by Gudyś *et al.* (2013). For more details on the feature set see Section S4 of Supplementary Material.

A stratified 10-fold cross-validation was run over five datasets against HuntMi (Gudyś *et al.*, 2013). The partitions were randomly constructed using a fixed seed in order to have fully reproducible experiments. Three datasets contained a mixture of sequences from different species, which were grouped into animals (10 species), plants (7 species), and viruses (29 species). The other two datasets contained sequences from *Homo sapiens* and *Arabidopsis thaliana*. The parameters used in miRNAss were the same in all the tests: $c = 1$ and $k = 10$ (for both RELIEF-F and graph construction). MiRNAss was tested with and without RELIEF-F, to measure its impact on

performance. This feature weighting was calculated for each fold, using the corresponding training partition.

The elapsed times measured¹ on each fold were averaged and are presented in Table 1. The first column shows the total number of sequences. The second and third columns indicate the number of pre-miRNA and non pre-miRNA sequences on each dataset. The fourth and fifth columns present the elapsed times for each method. The last column shows the times miRNAss was faster than HuntMi. The greatest difference was observed in the smallest dataset (virus): miRNAss was 38 times faster than HuntMi. The Arabidopsis is the second smallest dataset, and here miRNAss was 14 times faster. In the following three datasets, the computing time differences between miRNAss and HuntMi increased as the number of sequences increased. These results show that not only is miRNAss many times faster but also that the difference becomes greater as the number of sequences increases. Such differences are irrelevant in small datasets like the ones used in this subsection, but they can be very important in a real genome-wide prediction task, with several millions of sequences. For example, the time needed to train HuntMi with a 1.7 million sequences (one of the genome-wide datasets used in Section 3.3) can be estimated to be 37 days, whereas miRNAss only requires 18 hours.

Regarding prediction measures, both methods performed similarly in this experiment. A Friedman test was applied (Friedman, 1937), resulting in a p -value of 0.179. This demonstrates that the proposed method can obtain equivalent results to a state-of-the-art method for the supervised setup, although in less time (as shown in Table 1). Additionally, it was possible to verify that RELIEF-F improved the results on all tests, although in some of them the differences were small. This was expected, given that the features used in these datasets are the result of previous processes of feature selection. A Friedman test on this comparison results in a p -value of 0.025, which proves that the differences are significant. More details about these results can be seen in the Figure S5 in Supplementary Material. Another similar test with labeled data was conducted using scrambled pre-miRNAs as negative examples, since this could easily solve the problem of scarce negative examples for the supervised methods. The 1406 known pre-miRNAs from human were scrambled to create 1406 artificial non-pre-miRNA examples for training. Tests on human dataset had a low performance, with no significant differences for HuntMi and miRNAss (p -value = 0.2059).

To test the capacity of miRNAss to predict novel pre-miRNAs in different species, the well-known pre-miRNAs of animals and plants that were included in mirBase v17 were used for training, and the pre-miRNAs that were added in v18-19 were used for testing. In the case of animal species, microPred (Batuwita and Palade, 2009) was added for comparison purposes, since it was the best software for human miRNA prediction at the time of its publication. For plant species, the method added was PlantMiRNAPred (Xuan *et al.*, 2011), because it is specifically designed for plants. It should be noted that, again, the percentage of unlabeled sequences in the dataset is very low: less than 0.3% in animals and less than 1.3% in plants.

The S^+ obtained on each species is shown in Fig. 4, where each classifier is mapped with a different color across a radar plot. In the animals dataset, miRNAss outperformed microPred in almost all the species. In the human dataset, miRNAss obtained higher S^+ than microPred, which, as mentioned above, was specifically designed for humans. Compared with HuntMi, miRNAss obtained higher S^+ in 4 species, HuntMi produced better results in 4 species, and the results obtained were the same in 3 species. In plants, miRNAss outperformed PlantMiRNAPred in almost all the species. Compared with HuntMi, miRNAss had a better S^+ in 8 species, while HuntMi slightly outperformed miRNAss in only 2 species (*Cucumis melo* and *Sorghum bicolor*). To analyze if there

¹Intel®Core™i5-4460 CPU @ 3.20GHz, 8 GB of RAM.

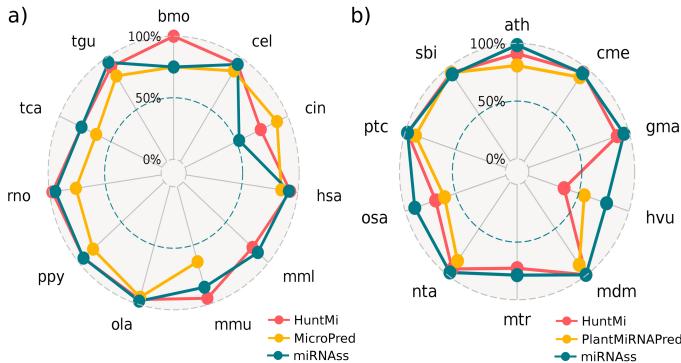


Figure 4: Sensitivity obtained with several state-of-the-art classifiers in: a) animal species, and b) plant species. The distance of each point to the center measures the sensitivity obtained on each species.

were significant differences between classifiers, Nemenyi tests (Nemenyi, 1962) were conducted for animals and plants datasets ($p < 0.05$). In animals species, HuntMi and miRNAss produced equivalent results, but both performed better than microPred. In plant species, miRNAss obtained the lowest rank (the best) and a significant difference with PlantMiRNAPred; however, the difference with HuntMi was not statistically significant.

Supervised methods make extensive use of labeled data, not only for training but also for finding the optimal hyper-parameters and thresholds. On the contrary, miRNAss was built over the hypothesis that labeled examples are scarce, unreliable, and not representative of the whole class, which is actually a more realistic scenario for this prediction task. Therefore, it should be noted that, even under these disadvantageous conditions, miRNAss obtained significantly better results than MicroPred and PlantMiRNAPred, and equivalent results to those produced by HuntMi, being, however, many times faster.

3.2 Few labeled examples

In order to depict a more realistic scenario, in which the number of known examples is very low, the five datasets used in the last tests were sampled in a train-test validation scheme with a varying percentage of labeled sequences. The percentage of labeled examples was reduced from 20% to 2%, with a step of 2%. The labeled examples were randomly selected and the tests were repeated 200 times for each percentage to estimate confidence intervals. In the Fig. 5, curves of expected F_1 with confidence intervals of 0.05 were estimated for comparison. In the human dataset, the F_1 is almost 10% higher for miRNAss, regardless of the percentage of labeled sequences. In the Arabidopsis dataset, where the number of positive examples sequences was lower, the differences in favor of miRNAss were the highest at the lower percentages of labeled examples, which indicates that miRNAss can effectively identify the positive class even with a very low number of labeled examples. The same trends are observed in the animals, plants, and virus datasets (see Supplementary Figure S6). These results not only show that miRNAss is capable of outperforming supervised methods when the number of labeled examples is low, but also that the error rates estimated using a high proportion of labeled examples are very different from those obtained in more realistic scenarios.

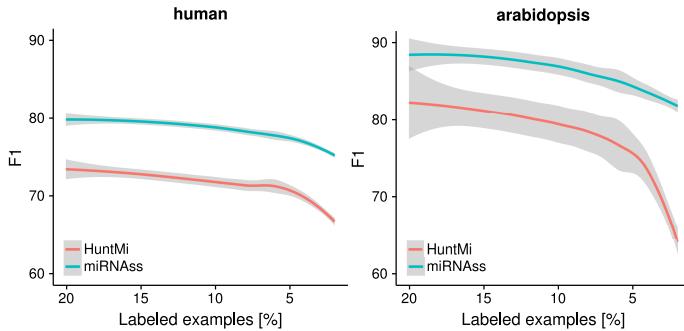


Figure 5: Curves of F_1 obtained by decreasing the percentage of labeled sequences. Shaded regions represent confidence intervals of the estimation with local regression (LOESS) at $p < 0.05$.

A further step towards a more realistic prediction task consists in using only positive examples. Under these conditions, miRNAss was compared with miRank (Xu *et al.*, 2008), which was designed to work with an extremely small number of positive examples. Datasets provided by the original author were used: 533 human pre-miRNAs and 1000 non-miRNA sequences. To make a fair comparison, both method used the feature set of miRank. This set is composed by 32 triplets (Xue *et al.*, 2005), normalized MFE, normalized base pairing propensities of both arms and normalized loop length. A varying number of positive examples (1, 2, 4, 8, 16, 32, 64, and 128) were labeled and the rest of the sequences were left unlabeled in order to measure the error rates. As the results depend on a random sampling of the sequences, this procedure was repeated 1000 times for each number of labeled examples. MiRank provides as output a continuous score; therefore, the prediction scores obtained with miRNAss were not thresholded. The area under the precision-recall curve was calculated (AUCPR), since it takes all possible thresholds into account (Bradley, 1997).

In this test, the 5% of the estimated n_- is automatically labeled by miRNAss as negative examples to initiate the algorithm. As it is shown in the Figure S1 of Supplementary Material, miRNAss is not sensitive to this parameter. Figure 6 presents a box plot with the distribution of AUCPR obtained by each classifier with a different number of labeled examples. MiRNAss maintained an almost constant AUCPR, irrespective of the number of labeled pre-miRNAs, while the performance of miRank decreased markedly. In addition, the values of AUCPR for miRNAss showed a little dispersion compared with miRank, which becomes very unstable when the number of labeled examples decreases. This instability may be produced by positive examples that are close to the frontier of the class, causing miRank to fail to set the decision boundary. The semi-supervised miRNAss algorithm correctly found the low-density regions that separate miRNAs from the rest of the sequences, regardless of the examples provided.

3.3 Real prediction in whole genomes

MiRNAss was tested with the genome-wide data of three well-known genomes: *Arabidopsis thaliana*, *Caenorhabditis elegans* and *Anopheles gambiae*, to reproduce all the conditions of a real prediction task. Previous works tested methods on genome-wide data from single species (Lai *et al.*, 2003; Bentwich *et al.*, 2005; Adai *et al.*, 2005; Huang *et al.*, 2007; Billoud *et al.*, 2014), but in this study we have processed three complete genomes and made all genome-wide datasets available for

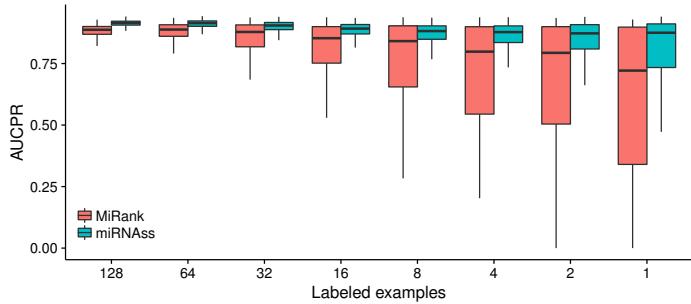


Figure 6: Box plot of the areas under the curve obtained by miRNAss and miRank, with a different number of positive training samples.

testing machine learning methods in pre-miRNAs prediction. The whole genomes were processed to extract all the existing stem-loops. For this purpose, all the chromosomes and mitochondrial genes were split in 600-nt-long windows, with a 100-nt overlap. The secondary structure of these sequences was predicted with RNAfold (Lorenz *et al.*, 2011). Then, all stem loops of at least 60 nt in length and 16 matched base pairs found in these structures were pruned and saved, taking care of deleting duplicated cases. This process left a total of 1,356,616 stem-loops of *A. thaliana*; 1,698,859 stem-loops of *C. elegans*; and 4,276,188 stem-loops of *A. gambiae*. BLAST (Camacho *et al.*, 2009) was used to match all the well-known pre-miRNAs annotated in mirBase v21 with the extracted stem-loops. This defined a total of 304, 249 and 66 hairpins as positive set for *A. thaliana*, *C. elegans*, *A. gambiae*, respectively. Two sets of features were extracted from the three genomes. The first one (FS1) is the same used in Section 3.1, to make a fair comparison with one of the methods. The second feature set (FS2) is an extended set composed by almost all features proposed for pre-miRNA prediction in literature. FS2 was calculated with miRNAfe (Yones *et al.*, 2015) and each feature vector resulted in 79 elements. For more details on the feature sets see Supplementary Material, Section S4. A 10-fold cross-validation (CV) scheme was used to measure miRNAss performance with averaged ROC curves.

Many pre-miRNA prediction algorithms were tested on these datasets to compare their performance with miRNAss. We have tried eleven predictors, but most of them failed with genome-wide data or their servers are not working. The only five predictors that could be used for these comparisons were HuntMi, Mirident (Liu *et al.*, 2012), HHMMiR (Kadri *et al.*, 2009), MiRPara (Wu *et al.*, 2011) and miR-BAG (Jha *et al.*, 2012). The first two methods produce hard class assignments (not scores), and that is why they are points in ROC figures. Instead, since MiRPara, miR-BAG, HHMMiR and miRNAss provide a score for each sequence, complete ROC curves can be drawn. MiR-BAG was not run on *A. thaliana* because it does not provide a model for plants. The output scores obtained with this method can only take four possible values, therefore, the ROC curves have straight lines. These results can be seen in Figure 7. In the *A. thaliana* genome, it can be seen that the miRNAss ROC curves are above miRPara and HHMMiR curves for all threshold values. It should be noted that Mirident, in spite of having the highest true positive rate (sensitivity), it also has the highest false positive rate. HuntMi is more balanced, having a high true positive recognition and a moderate number of false positives. Anyway, it is below miRNAss with both feature sets. In the *C. elegans* genome, a similar analysis can be done for HuntMi and Mirident.

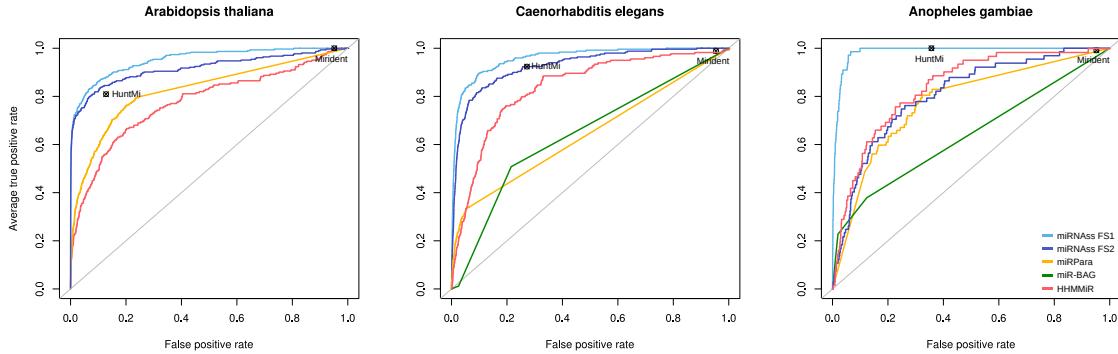


Figure 7: ROC curves for comparisons with state-of-the-art methods on genome-wide data from three species. The points show the performance achieved by methods that only return hard class assignments.

In this dataset, miR-BAG generates a ROC curve similar to the curve of MiRPara, both below the rest of the curves. HHMMiR presents a better performance than these methods, but again it is outperformed by miRNAss. In the case of the *A. gambiae* genome, performance of miRNAss with FS1 is more distant to the FS2. MiR-BAG and HHMMiR generate a curve similar to the obtained by miRNAss with FS2, far below the one obtained with FS1. The ROC curve with FS1 shows that, in the upper left corner, miRNAss can provide the best balance between sensitivity and false positive rate. In fact, this is nearly an ideal ROC curve.

Finally, as a summary of the comparative analysis, Table 2 presents more results of practical interest. The same methods and species of Figure 7 are here analyzed according to global performance and the total number of candidates that each method returns using their default threshold values, that is, the sum of true positives and false positives ($TPFP$). It can be seen that miRNAss outperforms all the methods in the three genomes. Mirident is the method with the lowest performance, for all species. This is because it labels as positive almost all examples, which is reflected in a very high sensitivity, but without practical utility given the number of candidates provided. MiR-BAG has a better but still poor performance in both species. HHMMiR and miRPara predict very few candidates, with high specificity at the cost of a very low sensitivity. HuntMi, instead, allows obtaining more balanced results, with the second best performance. However, for example in *A. gambiae*, it returns a number of false positives more than 5 times higher than those returned by miRNAss.

These results allow us to state that miRNAss outperforms supervised methods in a realistic classification setup. Artificially defined negative examples are used to train supervised models and, since these examples are not representative of the vast diversity of the negative class, the models fail to discard non-miRNA sequences correctly. By contrast, miRNAss can better take advantage of the very large number of unlabeled sequences to more tightly fit the decision boundary around the pre-miRNAs, discarding the rest of the sequences.

Table 2: Geometric mean of sensitivity and specificity (\bar{G}) and true+false positives ($TPFP$) on the three whole genome test.

Classifier	<i>A. thaliana</i>		<i>C. elegans</i>		<i>A. gambiae</i>	
	$TPFP$	\bar{G}	$TPFP$	\bar{G}	$TPFP$	\bar{G}
Mirident	1,294,648	22.05 %	1,617,221	21.29 %	4,068,431	21.86 %
miR-BAG	-	-	375,011	63.14 %	495,231	57.62 %
miRPara	2,755	47.95 %	11,712	53.79 %	283,232	72.48 %
HHMMiR	45,104	69.07 %	40,318	73.29 %	91,093	74.07 %
HuntMi	173,906	84.00 %	462,203	82.00 %	1,456,590	80.20 %
MiRNAss	134,369	84.82 %	164,557	87.61 %	258,096	93.34 %

4 Conclusions

In this study, we presented a new pre-miRNA prediction method called miRNAss, which uses a semi-supervised approach to face the problem of scarce and unreliable training samples. The experiments conducted in a forced supervised setup showed that miRNAss can achieve the classification rates of the best state-of-the-art methods in standard cross-validation tests, in shorter times. The proposed method was also tested under conditions that are closer to a real prediction task, where the number of labeled sequences is decreased. In these tests, miRNAss clearly outperformed the best available state-of-the-art supervised method, producing better results than a method that was specially designed to work under these conditions. The automatic search for negative examples proved to work well. The final test over all the stem-loops of three genomes using two different sets of features raises many important considerations. First of all, miRNAss widely outperforms the classification rates of supervised approaches. In addition, miRNAss proved to be efficient and scalable for handling over four million sequences. The results on this test also proved an important hypothesis made at the beginning of this study: the negative examples that are used to train many state-of-the-art prediction methods are not representative of the whole non-miRNA class. While those methods achieved very high error rates in cross-validation tests with an artificially defined negative class, the performance falls when they have to face the wide range of sequences that can be found in any genome. By contrast, miRNAss automatically searches for a wide variety of negative examples to initiate the algorithm. Then, miRNAss strives to take advantage of the distribution of unlabeled samples. As a result, it is capable of fitting tight decision boundaries around the pre-miRNAs using only a few positive examples.

Funding

This work was supported by Consejo Nacional de Investigaciones Científicas y Técnicas [PIP 2013 117], Universidad Nacional del Litoral [CAI+D 2011 548, 2016 082] and Agencia Nacional de Promoción Científica y Tecnológica [PICT 2014 2627], Argentina.

References

- Adai, A., Johnson, C., Mlotshwa, S., Archer-Evans, S., Manocha, V., Vance, V., and Sundaresan, V. (2005). Computational prediction of miRNAs in *arabidopsis thaliana*. *Genome research*, **15**(1), 78–91.
- An, J., Lai, J., Lehman, M. L., and Nelson, C. C. (2013). miRDeep*: an integrated application tool for miRNA identification from RNA sequencing data. *Nucleic acids research*, **41**(2), 727–737.
- Batuwita, R. and Palade, V. (2009). microPred: effective classification of pre-miRNAs for human miRNA gene prediction. *Bioinformatics*, **25**(8), 989–995.
- Bentwich, I., Avniel, A., Karov, Y., Aharonov, R., Gilad, S., Barad, O., Barzilai, A., Einat, P., Einav, U., Meiri, E., et al. (2005). Identification of hundreds of conserved and nonconserved human micrornas. *Nature genetics*, **37**(7), 766–770.
- Billoud, B., Nehr, Z., Le Bail, A., and Charrier, B. (2014). Computational prediction and experimental validation of micrornas in the brown alga *ectocarpus siliculosus*. *Nucleic acids research*, **42**(1), 417–429.
- Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern recognition*, **30**(7), 1145–1159.
- Camacho, C., Coulouris, G., Avagyan, V., Ma, N., Papadopoulos, J., Bealer, K., and Madden, T. L. (2009). BLAST+: architecture and applications. *BMC bioinformatics*, **10**(1), 1.
- Chapelle, O., Schölkopf, B., and Zien, A. (2006). *Semi-supervised Learning*. Adaptive computation and machine learning. MIT Press.
- Enright, A. J. and Ouzounis, C. A. (2001). Biolayout—an automatic graph layout algorithm for similarity visualization. *Bioinformatics*, **17**(9), 853–854.
- Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, **32**(200), 675–701.
- Gander, W., Golub, G. H., and von Matt, U. (1989). A constrained eigenvalue problem. *Linear Algebra and its applications*, **114**, 815–839.
- Gudyś, A., Szczęśniak, M. W., Sikora, M., and Makałowska, I. (2013). HuntMi: an efficient and taxon-specific approach in pre-miRNA identification. *BMC bioinformatics*, **14**(1), 83.
- Huang, T.-H., Fan, B., Rothschild, M. F., Hu, Z.-L., Li, K., and Zhao, S.-H. (2007). Mirfinder: an improved approach and software implementation for genome-wide fast microrna precursor scans. *BMC bioinformatics*, **8**(1), 341.
- Jha, A., Chauhan, R., Mehra, M., Singh, H. R., and Shankar, R. (2012). mir-bag: bagging based identification of microrna precursors. *PLoS One*, **7**(9), e45782.
- Joachims, T. et al. (2003). Transductive learning via spectral graph partitioning. In *ICML*, volume 3, pages 290–297.

- Kadri, S., Hinman, V., and Benos, P. V. (2009). Hhmmir: efficient de novo prediction of micrornas using hierarchical hidden markov models. *BMC bioinformatics*, **10**(1), S35.
- Kleftogiannis, D., Korfiati, A., Theofilatos, K., Likothanassis, S., Tsakalidis, A., and Mavroudi, S. (2013). Where we stand, where we are moving: surveying computational techniques for identifying miRNA genes and uncovering their regulatory role. *Journal of biomedical informatics*, **46**(3), 563–573.
- Kononenko, I. (1994). Estimating attributes: analysis and extensions of RELIEF. In *Machine Learning: ECML-94*, pages 171–182. Springer.
- Lai, E. C., Tomancak, P., Williams, R. W., and Rubin, G. M. (2003). Computational identification of drosophila microrna genes. *Genome biology*, **4**(7), R42.
- Liu, X., He, S., Skogerbo, G., Gong, F., and Chen, R. (2012). Integrated sequence-structure motifs suffice to identify microrna precursors. *PloS one*, **7**(3), e32797.
- Lopes, I. d. O., Schliep, A., and de Carvalho, A. C. d. L. (2014). The discriminant power of RNA features for pre-miRNA recognition. *BMC Bioinformatics*, **15**(1), 124.
- Lopes, I. d. O., Alexander, S., and de LF de Carvalho André P (2016). Automatic learning of pre-miRNAs from different species. *BMC Bioinformatics*, **17**(1), 224.
- Lorenz, R., Bernhart, S. H., Zu Siederdissen, C. H., Tafer, H., Flamm, C., Stadler, P. F., and Hofacker, I. L. (2011). ViennaRNA Package 2.0. *Algorithms for Molecular Biology*, **6**(1), 1.
- Malkov, Y., Ponomarenko, A., Logvinov, A., and Krylov, V. (2014). Approximate nearest neighbor algorithm based on navigable small world graphs. *Information Systems*, **45**, 61–68.
- Mease, D., Wyner, A. J., and Buja, A. (2007). Boosted classification trees and class probability/quantile estimation. *The Journal of Machine Learning Research*, **8**, 409–439.
- Nemenyi, P. (1962). Distribution-free multiple comparisons. In *Biometrics*, volume 18, page 263. INTERNATIONAL BIOMETRIC SOC 1441 I ST, NW, SUITE 700, WASHINGTON, DC 20005-2210.
- Ng, K. L. S. and Mishra, S. K. (2007). De novo SVM classification of precursor microRNAs from genomic pseudo hairpins using global and intrinsic folding measures. *Bioinformatics*, **23**(11), 1321–1330.
- Novák, P., Neumann, P., and Macas, J. (2010). Graph-based clustering and characterization of repetitive sequences in next-generation sequencing data. *BMC bioinformatics*, **11**(1), 378.
- Peace, R. J., Biggar, K. K., Storey, K. B., and Green, J. R. (2015). A framework for improving microRNA prediction in non-human genomes. *Nucleic acids research*, page gkv698.
- Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, **22**(8), 888–905.
- Wei, L., Liao, M., Gao, Y., Ji, R., He, Z., and Zou, Q. (2014). Improved and promising identification of human microRNAs by incorporating a high-quality negative set. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, **11**(1), 192–201.

- Wenyuan, L., Jing, M., Changwu, W., Baowen, W., and Yongqiang, L. (2013). The training set selection methods of microRNA precursors prediction based on machine learning approaches. In *Intelligent System Design and Engineering Applications (ISDEA), 2013 Third International Conference on*, pages 1566–1569. IEEE.
- Wettschereck, D., Aha, D. W., and Mohri, T. (1997). A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, **11**(1-5), 273–314.
- Wu, Y., Wei, B., Liu, H., Li, T., and Rayner, S. (2011). Mirpara: a svm-based software tool for prediction of most probable microrna coding regions in genome scale sequences. *BMC bioinformatics*, **12**(1), 107.
- Xu, Y., Zhou, X., and Zhang, W. (2008). MicroRNA prediction with a novel ranking algorithm based on random walks. *Bioinformatics*, **24**(13), i50–i58.
- Xuan, P., Guo, M., Liu, X., Huang, Y., Li, W., and Huang, Y. (2011). Plantmirnapred: efficient classification of real and pseudo plant pre-mirnas. *Bioinformatics*, **27**(10), 1368–1376.
- Xue, C., Li, F., He, T., Liu, G.-P., Li, Y., and Zhang, X. (2005). Classification of real and pseudo microrna precursors using local structure-sequence features and support vector machine. *BMC bioinformatics*, **6**(1), 310.
- Yones, C. A., Stegmayer, G., Kamenetzky, L., and Milone, D. H. (2015). miRNAsfe: a comprehensive tool for feature extraction in microRNA prediction. *Biosystems*, **138**, 1–5.

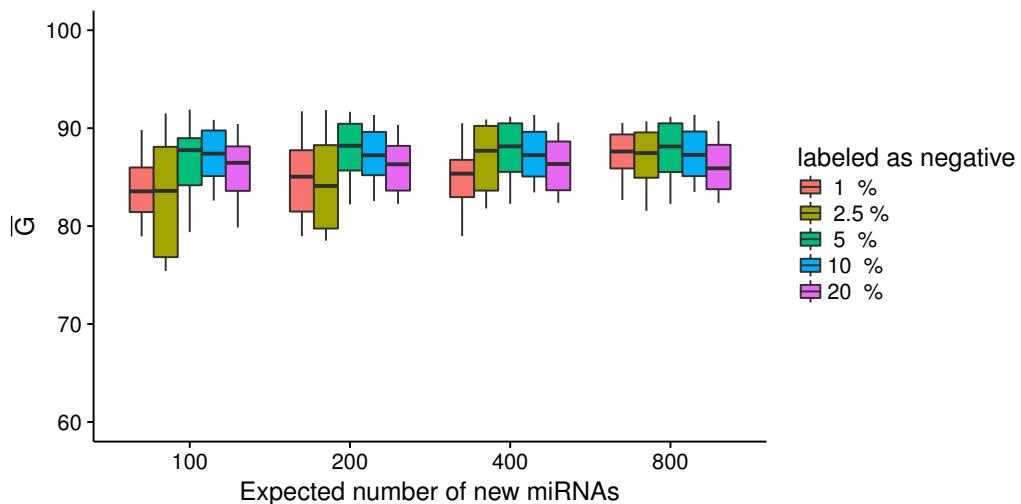
Supplementary Material

Genome-wide pre-miRNA discovery from few labeled examples

C. Yones, G. Stegmayer and D. H. Milone

Research Institute for Signals, Systems and Computational Intelligence, sinc(i), FICH-UNL,
CONICET, Santa Fe, Argentina.

Supplementary Figure S1: parameters sensitivity test



In this experiment, 20 combinations of the expected number of pre-miRNA to be find and the percentage of hairpins that will be labeled as negative examples to initiate the prediction algorithm have been tested on a cross-validation scheme, with the genome-wide dataset of *C. elegans*. The real number of true pre-miRNAs is 249, but as can be seen, for any estimation between 100 and 800 miRNAss maintains a good and stable performance. The same happens with the percentage of negative examples automatically labeled.

Supplementary Section S2: avoiding misclassification of labeled examples

To avoid the misclassification of positive examples, the constant c must be large enough to ensure that any misclassification of positive examples would yield a greater penalization than the regularizing term of the objective function. This value can be estimated from the equations of the method. Given the definition

$$a_{ij} = \begin{cases} \frac{\mu}{\mu + \|\mathbf{x}_i - \mathbf{x}_j\|^2} & \text{if } \mathbf{x}_j \in \mathcal{K}(\mathbf{x}_i) \text{ and } \ell_i \ell_j \geq 0 \\ 0 & \text{in other cases,} \end{cases} \quad (\text{S.1})$$

since the norm cannot be negative, then $a_{ij} \leq 1$. Therefore, $d_{ii} = \sum_{k=0}^n a_{ik} \leq k$. From (2) in the manuscript,

$$\begin{aligned} \mathbf{z}^T L \mathbf{z} &= \mathbf{z}^T I \mathbf{z} - \mathbf{z}^T D^{-1/2} A D^{-1/2} \mathbf{z} \\ &= \sum_i^n z_i^2 - \sum_i^n \sum_j^n \frac{z_i}{\sqrt{d_{ii}}} \frac{z_j}{\sqrt{d_{jj}}} a_{ij} \\ &= n - \sum_i^n \sum_j^n z_i z_j \frac{a_{ij}}{\sqrt{d_{ii}} \sqrt{d_{jj}}} \\ &\leq n - \sum_i^n \sum_j^n z_i z_j \frac{1}{k} \\ &= n - \sum_i^n \frac{z_i}{k} \sum_j^n z_j = n - \sum_i^n \frac{z_i}{k} 0 = n \end{aligned} \quad (\text{S.2})$$

Then, the misclassification of a positive example must have a penalization greater than n . A positive example \mathbf{x}_i is misclassified when $z_i \leq 0$, which leads to

$$c(z_i - \ell_i) C_{ii} (z_i - \ell_i) \geq c(0 - \ell_i) C_{ii} (0 - \ell_i) = c\gamma_+ C_{ii} \gamma_+. \quad (\text{S.3})$$

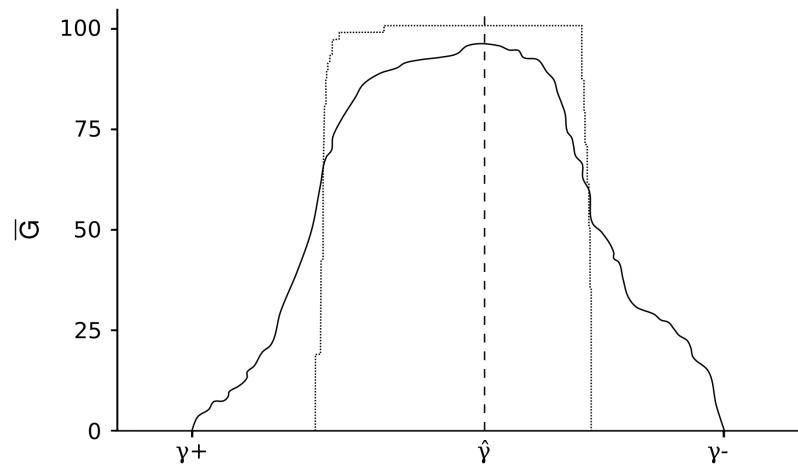
Then, to avoid the misclassification

$$\begin{aligned} c\gamma_+ C_{ii} \gamma_+ &> n \\ c\sqrt{\frac{n_-}{n_+}} C_{ii} \sqrt{\frac{n_-}{n_+}} &> n \\ c \frac{n_-}{n_+} C_{ii} &> n \\ cC_{ii} &> \frac{n_+ n_-}{n_-} \end{aligned} \quad (\text{S.4})$$

Therefore, any combination of c and C_{ii} that fulfills the inequality S.4 would avoid the misclassification of positive examples. One option is to leave $C_{ii} = 1$ as default, and set

$c > \frac{n_+n_-}{n_-}$. This parameter has the disadvantage that it also modifies the penalization for the negative examples. Another better solution is to leave $c = 1$ as default, and set $C_{ii} > \frac{n_+n_-}{n_-}$ for the positive examples. With this setting only the positive examples are protected from misclassification.

Supplementary Figure S3: thresholding the prediction scores



Comparison of the estimated (dotted line) and the real (solid line) geometric mean (\bar{G}) of sensitivity and specificity in an example dataset. Between two consecutive labeled samples in \mathbf{z}^* increasingly sorted by \bar{G} , there could be many unlabeled sequences. Hence, the estimated performance measure remains constant in those regions. When the number of labeled samples is low, this regions can be quite wide, therefore the final threshold ($\hat{\gamma}$) is set as the midpoint between the highest and the lowest scores in \mathbf{z}^* which maximizes the performance measure.

Supplementary Section S4: feature sets

Feature set 1 (FS1)

- tri_A , tri_U , tri_G , and tri_C : frequencies of secondary structure triplets composed of three adjacent nucleotides and the middle nucleotide: "A(((", "U(((", "G(((", and "C(((".
- orf : the maximal length of the amino acid string without stop codons found in three reading frames.
- $loops$: the cumulative size of internal loops found in the secondary structure.
- dm : a percentage of low complexity regions detected in the sequence using Dustmasker¹
- $\%C + G$: aggregated proportion of cytosine and guanine on the sequence.
- dG : Minimum free energy divided by the sequence length.
- dQ : is calculated as

$$\frac{1}{l} \sum_{i < j} p_{ij} \log_2 p_{ij},$$

where p_{ij} is the probability of pairing of nucleotides i and j . This value is calculated with the software RNAfold with -p option. Low values of dQ correspond to distributions dominated by a few bases likely to be matched. These bases are better predicted than those that have multiple alternative states.

- dF : topological descriptor. For a further description see Gan *et. al.* (1987)².
- $MFEI_1$: ratio between the minimum free energy and the $\%C + G$.
- $MFEI_2$: is calculated as dG/N_s , where N_s is the number of stems.
- $MFEI_3$: is calculated as dG/N_l , where N_l is the number of loops in the secondary structure.
- $MFEI_4$: is calculated as MFE/N_b , where N_b is the total number of base pairs in the secondary structure.

¹Morgulis, A., Gertz, E. M., Schaffer, A. A., & Agarwala, R. (2006). A fast and symmetric DUST implementation to mask low-complexity DNA sequences. *Journal of Computational Biology*, **13**(5), 1028-1040.

²Gan, H. H., Fera, D., Zorn, J., Shiffeldrim, N., Tang, M., Laserson, U., ... & Schlick, T. (1987). RAG: RNA-As-Graphs databaseconcepts, analysis, and features. *Nutrition and Health*, **5**(1-2), 1285-1291.

- zD : base pair distance for all pair of structures, calculated as

$$dD = \frac{1}{l} \sum_{i < j} p_{ij}(1 - p_{ij}),$$

normalized with z-score.

- *Diversity*: the structural diversity calculated with RNAfold (-p option).
- *NEFE*: Normalized Ensemble Free Energy calculated with RNAfold (-p option).
- *Diff*: is calculated as $|MFE - EFE|/L$.
- *dS*: Structure Entropy calculated using UNAfold.
- *dS/L*: normalized structure entropy.
- $|A - U|/L$, $|G - C|/L$, $|G - U|/L$: number of each possible base pair normalized by the sequence length.
- *Avg_BP_Stem*: Average of nucleotides per stem.
- $\%(A - U)/N_s$, $\%(G - C)/N_s$ and $\%(G - U)/N_s$: proportion of base pairs on stems.

Feature set 2 (FS2)

- Nucleotide proportion: ratio of each base in the sequence
- Dinucleotide proportion: ratio of dinucleotide elements of each kind.
- L : sequence length.
- N_s : number of stems.
- $\%C + G$: aggregated proportion of cytosine and guanine.
- G/C_{ratio} : ratio of guanine over cytosine.
- *Avg_BP_Stem*: average of nucleotides per stem.
- Longest stem length: longest region where the pairing is perfect.
- Terminal loop length: number of nucleotides in the stem region of the secondary structure.
- Number of base pair: number of paired nucleotides divided by 2
- dP : number of base pair divided by the nucleotide number.

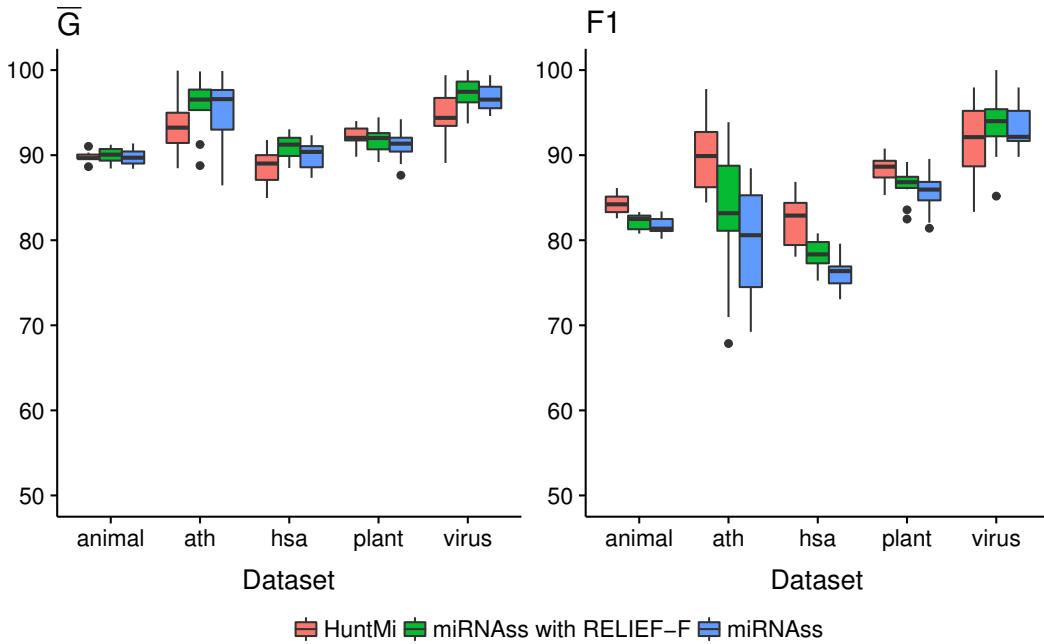
- $|A - U|/L$, $|G - C|/L$, $|G - U|/L$: number of each possible base pair normalized by the sequence length.
- $\%(A - U)/N_s$, $\%(G - C)/N_s$ and $\%(G - U)/N_s$: proportion of base pairs on stems.
- Triplets: Vector of 32 elements with the triplets frequency. A triplet is an element formed with the structure composition (paired or not paired) of three adjacent nucleotides and the base of the middle. An example of these elements is “.(A”, where the parenthesis represent a paired nucleotide, a dot a not paired one and the letter is the base of the middle nucleotide.
- MFE : minimum free energy.
- EFE : ensemble free energy.
- $Freq$: the structural frequency calculated with RNAfold (-p option).
- $Diversity$: the structural diversity calculated with RNAfold (-p option).
- $Diff$: is calculated as $|MFE - EFE|/L$.
- dG : Minimum free energy divided by the sequence length.
- dQ : is calculated as

$$\frac{1}{l} \sum_{i < j} p_{ij} \log_2 p_{ij},$$

where p_{ij} is the probability of pairing of nucleotides i and j . This value is calculated with the software RNAfold (-p option). Low values of dQ correspond to distributions dominated by a few bases likely to be matched. These bases are better predicted than those that have multiple alternative states.

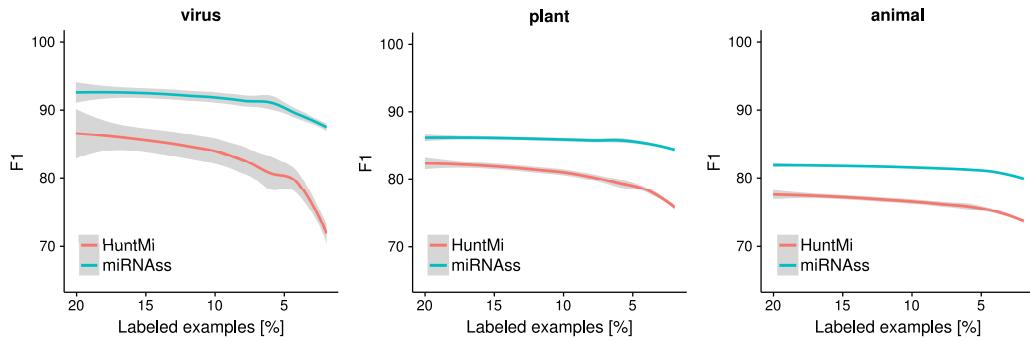
- $MFEI_1$: ratio between the minimum free energy and the $\%C + G$.
- $MFEI_2$: is calculated as dG/N_s , where N_s is the number of stems.
- $MFEI_4$: is calculated as MFE/N_b , where N_b is the total number of base pairs in the secondary structure.

Supplementary Figure S5: cross-validation in full labeled datasets



Box plot of \bar{G} and F_1 obtained by miRNAss and HuntMi in five different datasets. The middle line in each box represents the median of each distribution. The upper whisker extends from the hinge to the highest value, that is, within 1.5 times the interquartile range (IQR) of the hinge. The lower whisker extends from the hinge to the lowest value, within 1.5 times IQR of the hinge. Data beyond the end of the whiskers are outliers and are plotted as points.

Supplementary Figure S6: few labeled examples



Curves of F_1 obtained by decreasing the percentage of labeled sequences. The shaded regions are confidence intervals of the estimation with local regression (LOESS) at $p < 0.05$. In the Virus dataset, while the F_1 achieved by HuntMi falls as the percentage of labeled examples decreases, miRNAss maintains a higher and almost constant F_1 , independently of the percentage of labeled sequences. In the plant dataset, again, miRNAss achieves a higher F_1 for all percentages; and moreover, this difference increases as the number of labeled examples decreases. In animal dataset, miRNAss maintains an almost constant F_1 when the percentage of labeled sequences is greater than 5%. HuntMi achieves a lower F_1 for all percentages.

**Doctorado en Ingeniería
Mención en Inteligencia Computacional, Señales y Sistemas**

Título de la obra:

**Nuevo enfoque de aprendizaje
semi-supervisado para la identificación
de secuencias en biobinformática**

Autor: Cristian Ariel Yones

Lugar: Santa Fe, Argentina

Palabras Claves:

aprendizaje maquinal, aprendizaje semi-supervisado,
predicción de microRNA, genoma completo.