Reconocimiento de Patrones

Proyecto final

Aplicación de la teoría de la información a la selección de features para la clasificación de reads largos metagenómicos.

Guillermo Dufort y Álvarez

supervisado por
Sergio Martínez

Noviembre 2017

1 Introducción

1.1 Un poco de contexto sobre la metagenómica

La secuenciación del ADN es un conjunto de métodos y técnicas bioquímicas cuya finalidad es la determinación del orden de los nucleótidos (A, C, G y T) en una cadena de ADN. El conocimiento de las secuencias de ADN se ha vuelto indispensable para la investigación biológica básica, y en campos aplicados como el diagnóstico médico, la biotecnología, la biología forense, la virología y la sistemática biológica.

Tradicionalmente los estudios en la genómica se han concentrado en cultivar y secuenciar microbios de forma individual, y estudiar sus genomas. Este enfoque se torna problemático porque más del 99% de los microbios no pueden ser cultivados y por lo tanto sus genomas no pueden ser secuenciados. El campo de la metagenómica ha evolucionado para solucionar este problema. El objetivo de la metagenómica es tomar muestras ambientales que contienen múltiples organismos distintos en sus hábitats naturales, y luego, tomar estas mezclas y secuenciarlas utilizando técnicas de alto rendimiento.

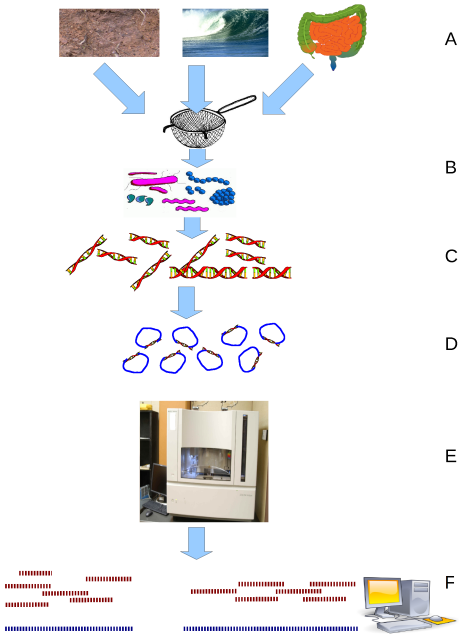


Figure 1: Secuenciamiento ambiental. (A) Muestreo del hábitat; (B) Filtrado de partículas, típicamente por tamaño; (C) Extracción de Lysis y de ADN; (D) Construcción de los clones; (E) Secuenciamiento de los clones; (F) Proceso de ensamblaje en genomas [1].

Estas técnicas se basan en cortar fragmentos grandes de ADN, para obtener otros más pequeños, que luego se secuencian completamente en tiras de bases denominadas *reads*, y se ensamblan computacionalmente en secuencias largas y contiguas, identificando las secuencias que se solapan entre sí. En la figura 1 se observa un esquema que resume los procesos que se realizan para realizar secuenciamiento de una muestra ambiental.

Uno de los desafíos que se presentan en la metagenómica es determinar qué especies se encuentran presentes en una muestra ambiental secuenciada. Este problema se traduce a realizar una clasificación taxonómica de los *reads* (las secuenciaciones de los pequeños fragmentos de ADN), y es el tema en el que se va a enfocar este trabajo.

1.2 Descripción del problema: Clasificación de reads

Los *reads* son básicamente strings formadas por las letras A, C, G, T, donde cada letra representa un base. Usualmente el largo de los reads, obtenidos con las tecnologías tradicionales, ronda en el entorno de las 500 bases. Si tenemos en cuenta que el largo del string que representa un genoma de una bacteria ronda cerca de las 50 millones de bases, es fácil ver que la tarea de determinar a qué organismo pertenece ese read no resulta trivial. Aún si queremos buscar una coincidencia exacta entre el read y un genoma de referencia, la tarea se vuelve más difícil sabiendo que los reads pueden contener errores, tanto de bases faltantes, bases de más, o bases equivocadas.

Entre los tipos de enfoque utilizados para enfrentar la clasificación de reads, el que estudiaremos en este proyecto es el supervisado, donde conocemos a priori cuáles son los posibles organismos de donde fueron secuenciados los reads, y conocemos los genomas de cada uno de estos organismos. Entre los métodos supervisados que existen, nos enfocaremos en métodos basados en composición que utilizan n-gramas, palabras de largo n, como features.

Los n-gramas son utilizados para construir perfiles de frecuencias (qué tan seguido ocurre una palabra en una secuencia dada), que luego son utilizados para construir modelos para los clasificadores. Por ejemplo, para los 3-gramas, dado que el alfabeto es de 4 letras, existen $4^3 = 64$ posibles tiras de largo 3, por lo que cada read se modela con 64 features, donde cada feature se corresponde con la cantidad de veces que aparece un cierto 3-grama en el read. Si tenemos el siguiente read,

$$AACAGGTAACGCGCTTAAGGCAC, \tag{1}$$

y utilizamos las frecuencias de los 2-gramas para modelarlo, entonces tenemos que los conteos para cada posible tira de las $4^2 = 16$ es,

$$\{AA:3, AC:3, AG:2, AT:0, CA:2, CC:0, CG:2, CT:1, GA:0, GC:3, GG:2,$$
$$GT:1, TA:2, TC:0, TG:0, TT:1\}.$$

El problema con esta representación es que los reads usualmente tienen distinto largo, por lo que para normalizar la representación, se divide cada frecuencia por la cantidad total de n-gramas en la tira, obteniendo así las probabilidades empíricas de cada n-grama en el read. Además, para hacer una representación vectorial del read, asumimos que las probabilidades empíricas se encuentran ordenadas en el orden lexicográfico de las tiras que tienen asociadas. De esta manera, podemos definir una función $\Phi_n : \{A, C, G, T\}^* \Rightarrow R^{4^n}$, que va desde el espacio de las strings de largo finito formadas por las letras $\{A, C, G, T\}$, en el espacio de features R^{4^n}, y se define como,

$$\Phi_n(r) = [\frac{\#s_1(r)}{largo(r) - (n-1)}, \frac{\#s_2(r)}{largo(r) - (n-1)}, ..., \frac{\#s_{4^n}(r)}{largo(r) - (n-1)}], \tag{2}$$

donde r es un read largo variable, $largo$ es una función que devuelve la cantidad de bases en el read, $\#s_i(r)$ es la cantidad de veces que aparece el n-grama s_i en r, y s_i es un n-grama que cumple

que $\forall j : 1 \leq j \leq 4^n$, si $j > i, s_j > s_i$ y si $j < i, s_j < s_i$, donde las comparaciones son por orden lexicográfico.

Dada la definición de Φ_n, podemos ver que a medida que crece n la dimensión del espacio de features crece exponencialmente, lo cual tiene múltiples lecturas. En principio, mientras mayor es la cantidad de features, mayores conflictos tendremos con la maldición de la dimensionalidad, siendo cada vez más propenso que al entrenar un clasificador, se sobreajuste la solución. Además, con valores de n relativamente pequeños, el problema se vuelve difícil de resolver en términos de complejidad (temporal y espacial), ya que con $n = 7$, cada read se representa con 16384 features.

Al mismo tiempo, es posible que con valores de n menores al largo de un read r perdamos mucha información al aplicarle la función Φ_n. Esto es fácil de ver si pensamos que para que la función $\Phi_n(r)$ sea biyectiva, n tiene que coincidir con el largo del read r. A medida que disminuye n respecto al largo de r, aumenta la cantidad de tiras que pueden generar $\Phi_n(r)$, lo que significa que menor es la información que conservamos del read original.

1.3 Objetivo del proyecto

Las dos lecturas presentadas en la sección anterior sobre n, nos dan pie a preguntarnos qué valor debe tomar n si queremos optimizar la efectividad de un clasificador, y al mismo tiempo mantener baja la complejidad del mismo. Con este problema en mente, es un objetivo de este trabajo estudiar cómo optimizar la capacidad de clasificación de reads de un clasificador, a través del uso de técnicas de selección y extracción de features. Específicamente, el objetivo es reproducir las técnicas de selección de features desarrolladas en [2], que utilizan la teoría de la información de forma estratégica para poder realizar clasificación taxonómica de reads. Además, otro objetivo de este trabajo es utilizar datos de reads obtenidos por una nueva tecnología llamada *Oxford nanopore MinION* [3], que produce reads de mayor largo al tradicional, con un promedio 4500 bases de largo. Los reads más largos tienen la desventaja de que tienen una mayor cantidad de errores que los tradicionales, pero en el contexto de la clasificación taxonómica pueden resultar ventajosos, ya que el cálculo de las probabilidades empíricas de ocurrencia de los n-gramas en el read puede ser más robusto al tener una mayor cantidad de n-gramas por read.

2 Descripción de los datos y preprocesaminto

2.1 Datos

Los datos que son utilizados para formar el conjunto de entrenamiento y el conjunto de test son reads que provienen de archivos en formato *FASTQ*[1]. Los reads son generados a partir de experimentos realizados con la tecnología *Oxford nanopore MinION*.

En el marco de este proyecto el objetivo que se propone es realizar una clasificación taxonómica entre 4 tipos distintos de bacterias (entre paréntesis se pone el sinónimo con el que serán referidas en el resto del documento): *bacillus anthracis* (baci), *escherichia coli* (coli), *enterobacter cloacae* (cloa), y *enterobacter raoultella* (raoul). Para construir el conjunto de entrenamiento se tomaron 4 archivos de secuenciación, cada uno de una bacteria distinta, y se obtuvieron 5000 reads de cada uno, por lo que el conjunto de entrenamiento cuenta con 20000 patrones divididos entre las 4 clases de forma balanceada. Por otro lado, el conjunto de test se construyó a partir de 4 archivos de secuenciación distintos, cada uno de una bacteria distinta, y se obtuvieron 1000 reads de cada uno, por lo que el conjunto de test cuenta con 4000 patrones divididos entre las 4 clases de forma balanceada.

Los reads obtenidos de cada archivo se toman de forma consecutiva, ya que, dada la naturaleza de los experimentos de secuenciación, no existe una correlación con respecto al genoma entre reads consecutivos (cada read es la secuenciación de un fragmento aleatorio del genoma). Además, los reads utilizados para el conjunto de entrenamiento y de test provienen de archivos distintos, con el objetivo de que los resultados no se encuentren sesgados a que reads de la misma clase pertenezcan al mismo experimento.

Además de los archivos de secuenciamiento, se cuenta con los genomas de referencia secuenciados para cada uno de los tipos de bacteria utilizados. Cada uno de los archivos tiene alrededor de 50 millones de bases de largo.

2.2 Preprocesamiento

Como fue explicado en la sección 1.2, el preprocesamiento de cada read para representarlo en el espacio de features se hace mediante la determinación de las probabilidades empíricas de los n-gramas en el read. Para cada n distinto, se tiene una representación del read distinta, determinada por la aplicación de la función Φ_n definida en (2), de manera que para cada n el espacio de features es R^{4^n}.

En el proyecto se trabajará con $n = 3, 4, 5$ y 6, con espacios de features de dimensiones $64, 256, 1024$ y 4096 respectivamente. Debido a restricciones en el hardware en donde se correrán los experimentos en este proyecto se utiliza como máximo n-gramas de largo 6.

[1] `https://en.wikipedia.org/wiki/FASTQ_format`

3 Primeros experimentos

Como una primera aproximación al problema se busca generar una intuición visual sobre la distribución, y separación, de los datos. Para eso se utilizó la herramienta *Weka*, utilizando PCA para extraer las dos features más descriminantes en el conjunto de training, y luego se graficó una en función de la otra, como se puede ver en la figura 2.

En la figura se puede observar claramente que para estas dos features las clases no son linealmente separables. Sin embargo, se puede ver que los patrones en rojo (baci) se encuentran agrupados y relativamente separados del resto, lo que da indicios de que se pueden obtener buenos resultados en su clasificación. Por otro lado, las otras tres clases se encuentran un poco más superpuestas, aunque la celeste (cloa) se encuentra un poco más separadas del resto. Tanto la clase azul (raoul) y la verde (coli), parecen estar bastante superpuestas entre sí y con las otras, pero mientras los patrones de coli (verde) parecen estar bien cercanos entre sí, los de raoul (azul) se encuentran más esparcidos, lo que puede indicar que su clasificación sea un poco más difícil.

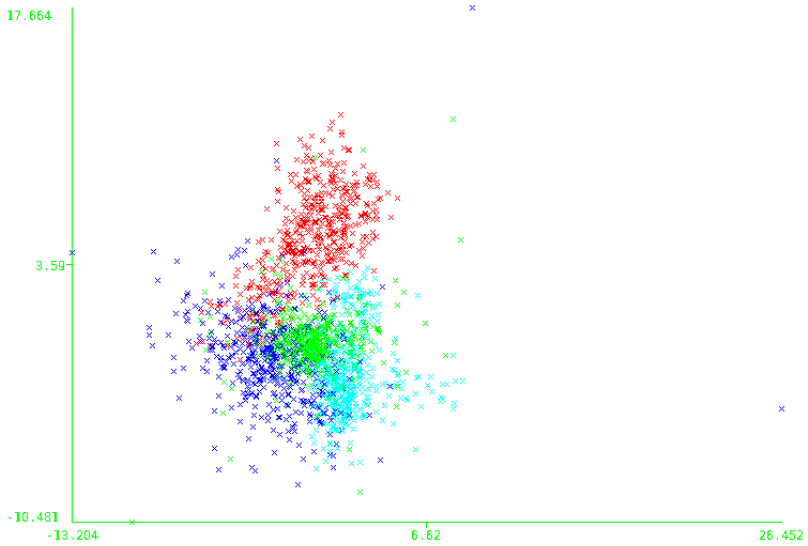
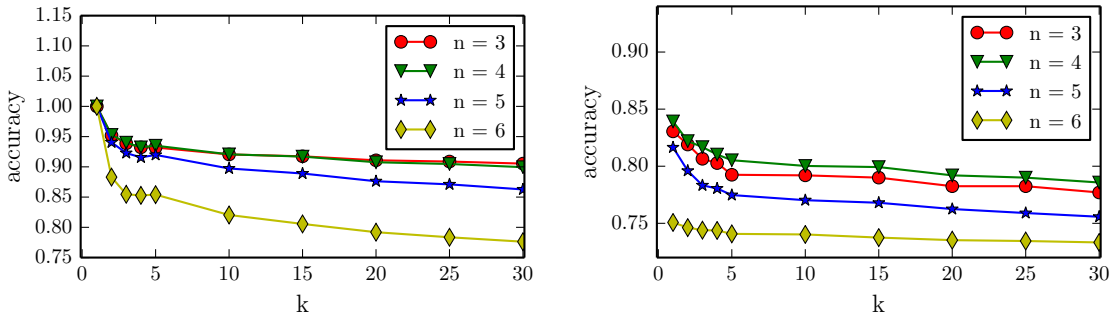


Figure 2: Representación visual en el plano de la distribución de las 4 clases de bacteria, tomando las 2 features más descriminantes mediante PCA y graficando una en función de la otra.

3.1 Prueba con clasificador basado en el algoritmo de vecinos más cercanos.

Como una primera aproximación a la clasificación de los datos, se entrenó un clasificador basado en el algoritmo de vecinos más cercanos, para diferentes valores de $k \in \{1, 2, 3, 4, 5, 10, 15, 20, 25, 30\}$ (cantidad de vecinos). Para la implementación del algoritmo se utilizó la biblioteca *sklearn* de PYTHON.

En las figuras 3a y 3b se pueden observar los resultados en términos de accuracy para los diferentes valores de k en los distintos valores de n, tanto para el conjunto de training como el conjunto de testing. Los resultados muestran que el mejor resultado de accuracy sobre el conjunto de testing



(a) Resultado sobre datos de *train*.

(b) Resultados sobre datos de *test*.

Figure 3: Resultados de la clasificación de un clasificador knn, entrenado sobre el conjunto de training, para diferentes k y n.

se logra con $n = 4$, y se alcanza 84% de acierto. En principio parece un resultado un poco extraño, ya que los valores más bajos de k sobreajustan la solución a los datos de entrenamiento, lo cual, en la mayoría de los problemas, esto deriva en que la generalización, osea las pruebas sobre test, de malos resultados. Sin embargo, en este problema, parece ser una constante que para todos los n los resultados en términos de accuracy empeoran a medida que k aumenta. Este resultado parece ser un fuerte indicador de que el problema al que nos enfrentamos requiere de una frontera de decisión compleja.

Es interesante ver la matriz de confusión generada para los parámetros que alcanzan los mejores resultados, $n = 3$ y $k = 1$. En el caso del conjunto de training, por la naturaleza del algoritmo de vecinos más cercanos, sabemos que si todas las muestras son distintas (como es este caso), el acierto sobre el conjunto de training del clasificador knn es 100%, por lo que no tiene sentido ver la matriz de confusión.

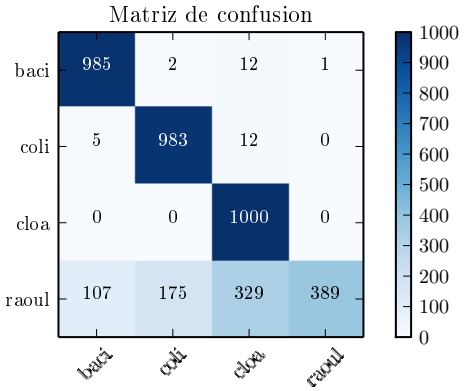


Figure 4: Matriz de confusión para el clasificador knn con parámetro $k = 1$, sobre $n = 4$, sobre el conjunto de testing.

En la figura 4 se puede observar que para las clases, baci, coli y cloa, los resultados son excelentes, alcanzando niveles de recall casi perfectos, pero sin embargo el nivel de precisión se ve afectado por la mala clasificación de los patrones de la clase raoul. En cierta medida, este resultado se podía anticipar al observar que la distribución espacial en el gráfico 2 de la clase raoul estaba significativamente entreverada con las otras clases. Estos resultados nos indican que lograr una buena clasificación de la clase raoul será clave para lograr una buena clasificación general.

Por otro lado, volviendo a la figura 3, es importante notar que mientras que los resultados para $n = 3$ y $n = 4$ son relativamente similares, para $n = 5$ y, sobretodo para $n = 6$, los niveles de acierto empeoran significativamente. Esto indica que a medida que aumenta el tamaño del espacio de las features el algoritmo knn logra cada vez peores resultados.

3.2 Prueba con clasificador SVM-C y kernel *rbf*.

El clasificador knn puede lograr muy buenos resultados, pero en términos de eficiencia computacional es muy costoso, por lo que usualmente es utilizado para realizar pruebas sobre los datos, pero no para llevarlo a implementaciones reales. Con el objetivo de utilizar algoritmos más eficientes, y siguiendo los pasos del trabajo de referencia en [2], se propone utilizar SVM-C como algoritmo de clasificación. Además, en [2] se explica que en el contexto de la clasificación taxonómica de reads representados con frecuencias de n-gramas, tradicionalmente el algoritmo SVM-C ha obtenido buenos resultados.

Para realizar las pruebas con SVM-C se utilizó el clasificador *SVC* del paquete *sklearn* de PYTHON. Además, se decidió utilizar el kernel *rbf* debido a su gran capacidad de crear fronteras de decisión de alta complejidad, que en el contexto del problema, a partir de los análisis del algoritmo de vecinos más cercanos, vemos que pueden resultar necesarias. Como el problema en cuestión es multiclase, se utilizó la extensión de SVM a multiclase "one to one", que es la que utiliza SVC por defecto.

El algoritmo *SVC* recibe como parámetros principales C y γ; C es una variable que ajusta el peso que se le dan a los errores de clasificación, de manera que a medida que aumenta C el algoritmo prioriza clasificar todas las muestras de manera correcta, dándole al modelo la libertad de elegir más vectores de soporte. Por otro lado, intuitivamente, γ define el alcance de la influencia de los vectores de soporte, con valores bajos significando "lejos" y valores altos "cerca". El comportamiento del modelo es muy sensible al valor del parámetro γ. Si γ es muy grande, el radio del área de influencia del vector de soporte sólo incluye al vector mismo, y ningún cambio en el parámetro C puede evitar que se sobreajuste el modelo. Cuando γ es muy pequeño, el modelo está muy restringido y no puede capturar la complejidad de los datos. Con la información que manejamos hasta el momento, podemos estimar que vamos a necesitar de un γ relativamente alto, ya que las pruebas con knn nos indican que el sobreajuste en el conjunto de training lleva a buenos resultados.

Para estimar los valores de C y γ que mejor clasifican los datos se realiza *gridsearch* con la función *GridSearchCV* provista por *sklearn*. Para cada $n \in 3, 4, 5, 6$, se calculó cuál combinación de C y $\gamma \in \{0.01, 0.1, 1, 10, 50, 100, 1000\}$ da los mejores resultados de precisión, y cuál combinación da los mejores resultados de recall. Cada uno de los experimentos se corrió sobre el conjunto de training, con *cross validation* en 5 iteraciones, utilizando el 50% del conjunto de training para entrenar y lo restante para evaluar.

Debido a la gran cantidad de resultados obtenidos, en el documento se reporta únicamente la pareja de valores que alcanzan los mejores valores de precisión y recall, que son $C = 10$ y $\gamma = 100$.

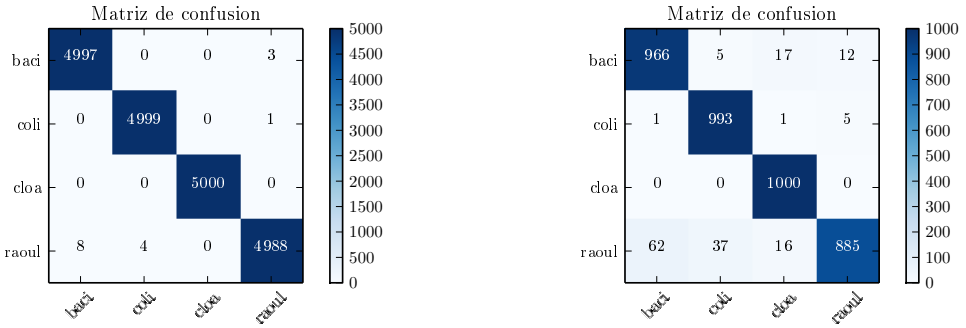
Estos valores de los parámetros son los que para la mayoría de los n alcanzan tanto el mejor valor de precisión como de recall. En algunos casos, el mejor valor de γ es 50, pero dado que la diferencia en performance es de algunas centésimas respecto a $\gamma = 100$, se decidió establecer como óptimo $\gamma = 100$ para todos los n.

n	Accuracy training (%)	Accuracy testing (%)
3	97,4	89,3
4	99,2	93,1
5	99,7	95,0
6	**99,9**	**96,1**

Figure 5: Resultados en términos de accuracy sobre el conjunto de training y el conjunto de testing, para el clasificador SVM-C con $C = 10$ y $\gamma = 100$, para los diferentes n.

En la tabla presentada en la figura 5 se pueden ver los resultados obtenidos con los parámetros óptimos para los diferentes n. A diferencia del clasificador knn, los resultados en términos de accuracy mejoran a medida que aumenta n y crece la dimensionalidad del espacio de features. De forma similar al clasificador knn, a medida que existe un mayor sobreajuste a los datos de entrenamiento, generando clasificadores con accuracy en training cercano al 100%, los resultados sobre el conjunto de test mejoran. Esto es consistente con que el mejor parámetro para knn fuera 1, y con que el mejor γ encontrado sea 100, un valor alto que produce una frontera de decisión compleja ajustada a los datos.

Específicamente, para $n = 6$ se logran los mejores resultados, alcanzando un $96,1\%$ de accuracy, con un clasificador que tiene 4086 vectores de soporte.



(a) Resultado sobre datos de *train*.

(b) Resultados sobre datos de *test*.

Figure 6: Matrices de confusión para el algoritmo SVM-C con parámetro $C = 10$, $\gamma = 100$, y $n = 6$.

En la figura 6 se pueden ver las matrices de confusión para $n = 6$ con los parámetros óptimos. En sintonía con los resultados previos, la clase raoul sigue siendo la que presenta mayores dificultades para ser clasificada, pero los resultados se logran mejorar significativamente.

3.3 Análisis de resultados

Los resultados obtenidos con knn y con SVM-C, indican que para clasificar de forma correcta este tipo de datos, se precisa de fronteras de decisión de alta complejidad. Específicamente, con SVM-C se logran los mejores resultados en $n = 6$, osea donde la dimensionalidad del espacio de features es la mayor, con 4096 features por read.

Lamentablemente, por restricciones de la capacidad del hardware utilizado, no se pudieron realizar pruebas con n-gramas de mayor tamaño, que a partir de los resultados obtenidos, suponemos que pueden mejorar aun más los resultados. Este fenómeno puede estar directamente relacionado con el hecho de que a medida que aumenta n, también aumenta la cantidad de información que tiene cada vector de features correspondiente a un read. Esto lo podemos ver claramente, en el hecho de que si tenemos la representación de un read con las probabilidades condicionales empíricas de los $(n+1)$-gramas, podemos reconstruir las probabilidades condicionales empíricas de los n-gramas. Osea que los $(n+1)$-gramas tienen al menos la información de los n-gramas.

A pesar de que los mejores resultados se logran con $n = 6$, en términos de eficiencia, el clasificador SVM-C entrenado con patrones de 4096 features es el más complejo y por lo tanto el menos eficiente en términos de tiempo. Nos planteamos entonces como objetivo del resto del documento, intentar lograr resultados similares a los mejores obtenidos por SVM-C con parámetros óptimos (accuracy 96,1%), pero en un espacio de features de dimensión significativamente menor, a través del uso de técnicas de selección y extracción de features.

4 Selección de features

En esta sección se exploran los resultados de aplicar dos técnicas de selección de features para construir nuevos clasificadores.

Por un lado se utiliza un método de selección provisto por la libraría *sklearn* llamado *SelectKBest*, que evalúa cada feature mediante la función chi cuadrado. La función chi cuadrado intenta medir la dependencia entre cada una de las features, y las clases, de manera que las que tienen más probabilidad de ser independientes son descartadas, ya que pueden ser irrelevantes para la clasificación. Para evaluar el funcionamiento de *KBest*, primero se transforma el espacio de features a las k mejores seleccionadas, y luego se utiliza el clasificador SVM-C con parámetros optimizados encontrado en la sección anterior.

Por otro lado, se construye manualmente un método de selección, y un nuevo clasificador, basado en la Teoría de la Información (TI), reproduciendo lo efectuado en [2].

4.1 Selección de features basada en TI

Uno de los objetivos tradicionales de la teoría de la información es maximizar la capacidad de un canal, preservando las partes de una señal más informativas, y simultáneamente ignorando las menos informativas. Esta herramienta nos puede ayudar en la clasificación de reads si la utilizamos para detectar qué features son las que maximizan nuestra habilidad de distinguir entre genomas, ignorando las features que ofrecen menos información en este aspecto.

4.1.1 Funciones de ranking

Basándonos en [2], para seleccionar cuáles n-gramas son los que mejor permiten distinguir entre genomas, se utilizaron tres funciones de ranking de features: la divergencia simétrica de Kullback-Leibler (KL), la información mutua (IM), y la diferencia absoluta entre probabilidades empíricas de n-gramas de dos genomas. Las tres funciones utilizan las probabilidades empíricas de los n-gramas en los genomas de cada bacteria. Esto quiere decir que, primero, dado un cierto n, se calculan para cada clase de bacteria, las probabilidades empíricas de cada n-grama sobre el *genoma de referencia* de la bacteria, aplicando la función Φ_n sobre el genoma. Es importante notar que el método está completamente basado en información a priori y no utiliza información específica obtenida del conjunto de entrenamiento, como sí lo hace *KBest*.

El método planteado rankea las features más relevantes de a pares de clases. Osea, que en nuestro caso que tenemos 4 clases distintas, existen $\binom{4}{2} = 6$ pares distintos, y se tiene un ranking de features para cada uno de estos pares.

El objetivo de las funciones de ranking r es que, dadas $p_A(X)$ y $p_B(X)$, las probabilidades empíricas del n-grama X en los genomas A y B, $r(p_A(X), p_B(X))$ devuelve un valor que representa la relevancia que tiene el n-grama X en la clasificación de reads entre los genomas A y B. De esta manera, si calculamos $r(p_A(X), p_B(X))$ para todos los n-gramas, y los ordenamos de mayor a menor, tenemos el ranking de features que buscamos.

La primer función de ranking utilizada es la Divergencia simétrica de Kullback-Leibler,

$$r_{KL}(p_A(X), p_B(X)) = p_A(X) \log \frac{p_A(X)}{p_B(X)} + p_B(X) \log \frac{p_B(X)}{p_A(X)}. \quad (3)$$

Luego se utiliza como función de ranking la información mutua del n-grama con el conjunto de clases,

$$r_{IM}(p_A(X), p_B(X)) = p_A(X) \log \frac{p_A(X)}{p_C(X)} + p_B(X) \log \frac{p_B(X)}{p_C(X)}, \quad (4)$$

donde $p_C(X) = \frac{\#X_A + \#X_B}{\sum_X \#X_A + \sum_X \#X_B}$, siendo $\#X_A$, y $\#X_B$, la cantidad de ocurrencias del n-grama X en el genoma A y el genoma B, respectivamente. Osea, que básicamente $p_C(X)$ es la probabilidad empírica total de ocurrencia del n-grama X en los dos genomas.

Por último, se utiliza como función de rankeo la diferencia entre las probabilidades empíricas de los n-gramas,

$$r_{IM}(p_A(X), p_B(X)) = |p_A(X) - p_B(X)|. \quad (5)$$

Las tres funciones planteadas buscan asignar a cada n-grama un valor que represente su relevancia a la hora de distinguir entre dos clases. Esto lo logran utilizando el principio básico de que si las probabilidades empíricas $p_A(X)$ y $p_B(X)$ de un n-grama X en cada genoma A y B respectivamente, son similares, entonces su relevancia a la hora de clasificar es baja.

4.1.2 Creación del ranking unificado

Utilizando las funciones de ranking planteadas en la sección anterior se generan 3 rankings para cada par de clases. Para poder unificar estos tres rankings en uno, lo que se hace es normalizarlos y luego hacer uno nuevo, en donde la relevancia de cada feature es la suma de las relevancias en cada uno de los rankings normalizados. Luego, si queremos obtener las k features más relevantes, se seleccionan las primeras k features del ranking unificado, ordenado de mayor a menor. En la figura 7 se muestra el flujo del algoritmo desarrollado para seleccionar las k features más relevantes.

4.1.3 Clasificador SVM-C con selección de features basada en TI

Como vimos en la sección anterior, el método desarrollado para selección de features basada en TI nos da un ranking de las features más relevantes por cada par de clases de bacteria. Para tomar ventaja de esto, se utiliza el método de clasificación multiclase de SVM "one to one".

Este método construye un clasificador SVM-C para cada par de clases, con los parámetros óptimos determinados anteriormente, y los entrena en el conjunto de training. Luego para clasificar una nueva muestra, a cada clasificador SVM creado se le da un voto, y por mayoría se decide la clasificación de la muestra. En caso de haber un empate de votos, se utilizan las probabilidades que arroja el algoritmo SVM-C, de que la clasificación sea correcta. Estas probabilidades se calculan utilizando la posicion de la muestra respecto al margen determinado por el algoritmo SVM-C.

Para combinar la selección de features basada en TI y el clasificador SVM-C "one to one" lo que se hace es realizar una selección de las k features más relevantes para cada par de clases, y luego se entrena el clasificador respectivo del par de clases en el espacio de features reducido.

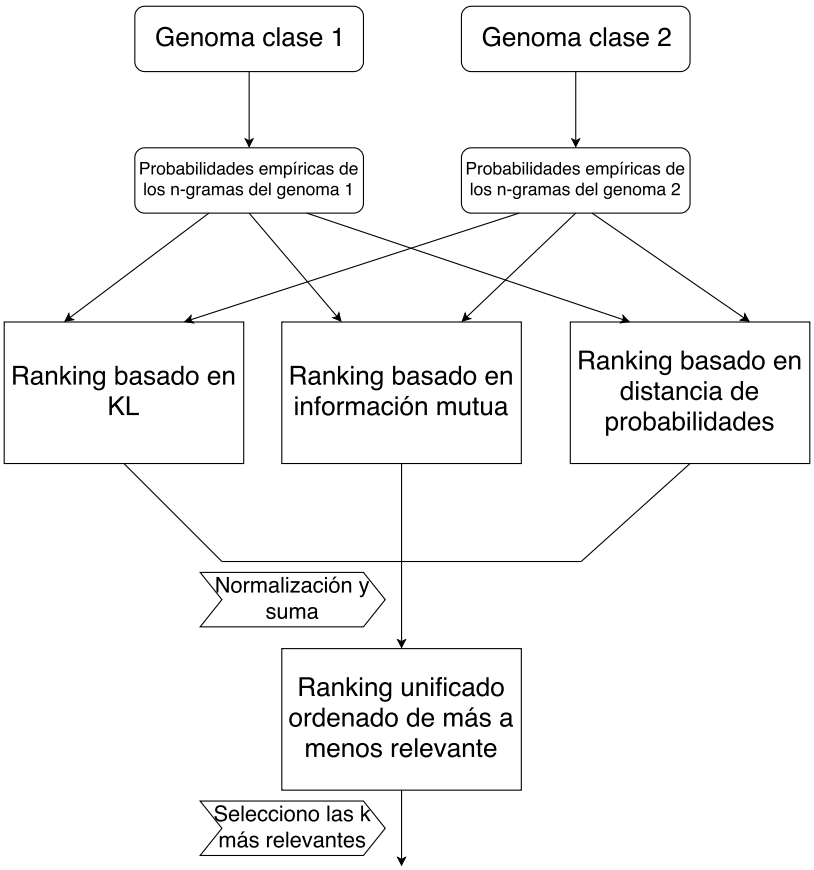


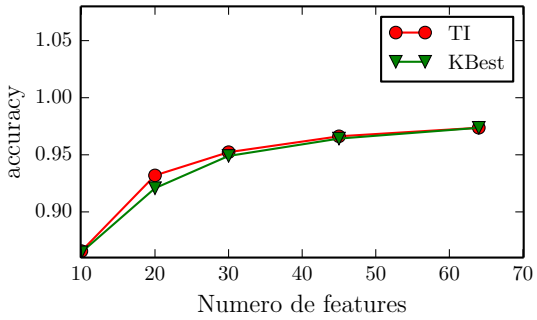
Figure 7: Flujo del algoritmo basado en teoría de la información para la selección de k features.

4.2 Pruebas y análisis de resultados

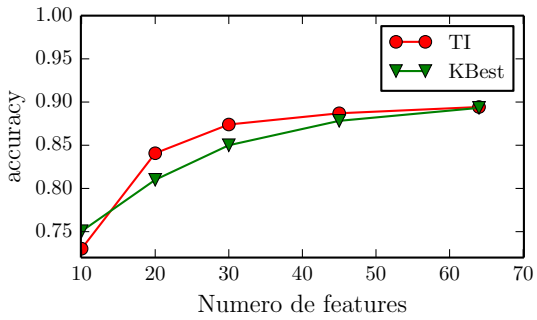
Para evaluar la performance del clasificador propuesto, se hicieron pruebas sobre todos los posibles n, eligiendo diferentes valores de k para cada n. Para comparar la performance del método de selección de features propuesto se lo compara con *KBest*. En las figuras 8, 9, 10, y 11, se pueden observar los resultados obtenidos.

En principio es interesante observar que a excepción de $n = 3$, el método *KBest* logra en general mejores o iguales resultados que TI sobre el conjunto de training. Esto tiene sentido, ya que *KBest* se ajusta directamente sobre el conjunto de training, mientras que TI elige las features más relevantes observando los genomas de referencia (información a priori). La hipótesis a paritr de esto, es que el método TI debería poder capturar información más general, y por ende, funcionar mejor sobre datos no vistos. Si vemos los resultados obtenidos sobre el conjunto de test, podemos ver que, efectivamente, se presenta este comportamiento.

El método TI en casi todas las pruebas sobre el conjunto de testing, funciona mejor que el método *KBest*. Sin embargo, la diferencia se reduce a medida que incrementa el n, y esto puede estar directamente relacionado con el hecho de que para n más grandes, cada read contiene más información.

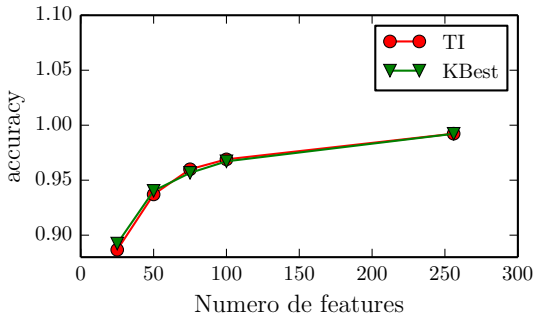


(a) Resultado sobre datos de *train*.

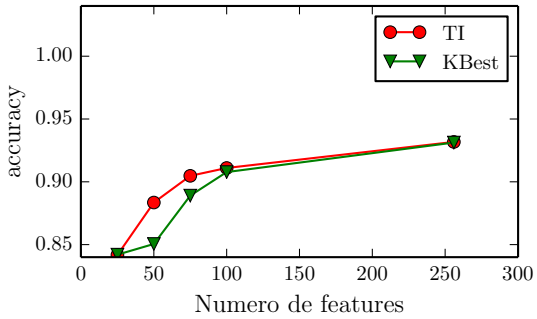


(b) Resultados sobre datos de *test*.

Figure 8: Resultados de la clasificación de un clasificador SVM-C con selección de features basado en TI, sobre n=3, en comparación con SVM-C con selección de *KBest*.



(a) Resultado sobre datos de *train*.



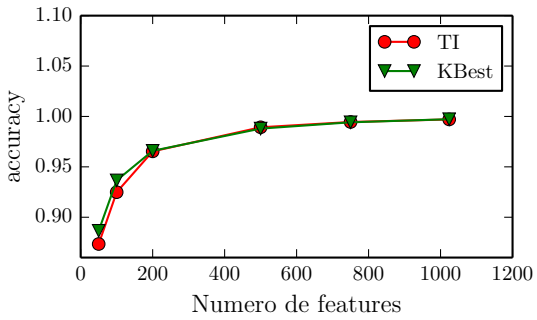
(b) Resultados sobre datos de *test*.

Figure 9: Resultados de la clasificación de un clasificador SVM-C con selección de features basado en TI, sobre n=4, en comparación con SVM-C con selección de *KBest*.

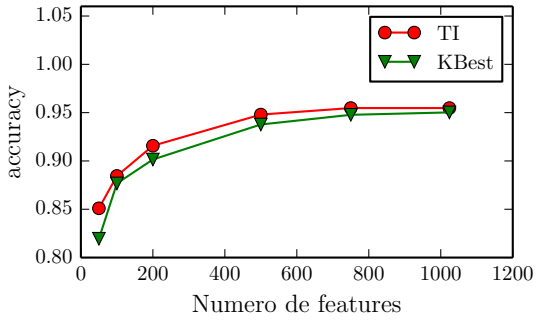
De esta manera, *KBest* cuenta con mayor información y realiza menos sobreajuste sobre conjunto de entrenamiento, equiparando cada vez más la performance del método TI.

En términos de accuracy, el fenómeno que se observa en todas las gráficas es de un rápido crecimiento de accuracy a medida que aumentan las features, y luego el crecimiento se estabiliza alcanzando una pendiente casi horizontal. Este fenómeno es más notorio para los n que tienen mayor cantidad de features como $n = 5$ y $n = 6$. Para $n = 5$ se puede observar que el crecimiento a partir de las 500 features se vuelve menor, y para $n = 6$ este fenómeno se puede observar de las 1000 features en adelante. Esto nos está diciendo que para este problema de clasificación existe un espacio de features de menor dimensión, que concentra la mayor cantidad de información, y que por lo pronto podrían existir features que no aportan nada más que ruido.

Por otro lado, otra lectura interesante, es que si queremos lograr buenos resultados con muy pocas features, digamos 50 o 100, se deben seleccionar las features de espacios dimensionales más

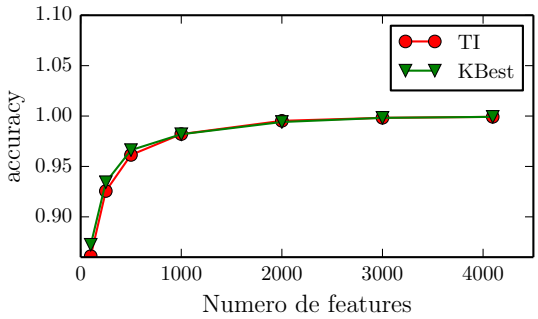


(a) Resultado sobre datos de *train*.

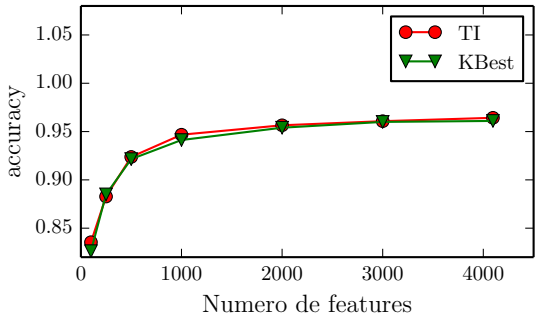


(b) Resultados sobre datos de *test*.

Figure 10: Resultados de la clasificación de un clasificador SVM-C con selección de features basado en TI, sobre n=5, en comparación con SVM-C con selección de *KBest*.



(a) Resultado sobre datos de *train*.



(b) Resultados sobre datos de *test*.

Figure 11: Resultados de la clasificación de un clasificador SVM-C con selección de features basado en TI, sobre n=6, en comparación con SVM-C con selección de *KBest*.

pequeños. La información en los espacios de dimensión menor se encuentra más concentrada en cada feature, de manera que realizar una selección de pocas features en un espacio de dimensión menor que otro, nos permite obtener representaciones de los reads con más información. Esto se puede ver, por ejemplo, en que el resultado de accuracy que se obtiene para TI con $n = 5$ y $k = 100$ es 88,5%, mientras que para $n = 6$ es 83,6%.

5 Extracción de Features

Con el objetivo de lograr mejorar los resultados en espacios de baja dimensionalidad de features, se prueban técnicas de extracción de features, específicamente PCA. La implementación utilizada de PCA es la que provee la biblioteca de PYTHON *sklearn*, con los parámetros por defecto.

La idea es utilizar el método de selección de features propuesto para seleccionar las features que acumulan la mayor cantidad de información, descartando las que sólo agregan ruido, y luego utilizar la extracción de features para concentrar esa información en un conjunto de features reducido.

En la figura 12 se muestran los resultados para $n = 5$ y $n = 6$, de utilizar PCA eligiendo las 50, y las 100, features que acumulan mayor varianza, sobre el espacio total de features, y sobre el espacio reducido mediante el método de selección de features TI. Para $n = 5$ se seleccionaron las 750 features más relevantes, y para $n = 6$ las 3000 más relevantes, en ambos casos es alrededor de 3/4 del espacio de features.

n	pca features	SVC+pca (Acc Test %)	**k**	SVC+TI+pca (Acc test %)
5	50	92,7	750	90,5
	100	93,4	750	**94,4**
6	50	94,3	3000	91,6
	100	95,1	3000	**95,4**

Figure 12: Resultados en términos de accuracy sobre el conjunto de testing para el clasificador SVM-C utilizando PCA y utilizando selección de features TI más PCA.

Los resultados muestran que PCA en general logra muy buenos valores de accuracy sobre el conjunto de test, alcanzando un máximo de 95,1% de accuracy en un espacio de features de dimensión 100. Este resultado es tan solo un 1% peor que el alcanzado por SVM-C sobre el espacio de 4096 features. Lo más interesante es que al aplicar el algoritmo de selección de features TI, y luego pca con 100 features, tanto para $n = 5$ como $n = 100$, los resultados alcanzados son mejores que los que se alcanzan utilizando únicamente PCA. Esto puede ser un indicador de que el método de selección de features efectivamente realiza una "limpieza" del espacio de features, descartando las que no tienen valor para realizar clasificación, y que además introducen ruido que luego hace que la clasificación empeore.

6 Conclusiones y trabajo futuro

Como objetivo de este proyecto nos planteamos estudiar cómo optimizar la capacidad de clasificación de reads de un clasificador, a través del uso de técnicas de selección y extracción de features. Para esto, reprodujimos las técnicas de selección de features desarrolladas en [2], que utilizan la teoría de la información, y además realizamos pruebas con PCA.

Con respecto a la selección de features, comparamos el método TI desarrollado, con *KBest*, llegando a la conclusión de que para las representaciones de los reads en espacios de features de menor dimensión, el método desarrollado funciona significativamente mejor que *KBest*. Sin embargo, a medida que la dimensión del espacio es mayor, y existe más información en cada vector de features, la información a priori utilizada por el método TI pierde relevancia y *KBest* se comporta de forma similar a TI.

Por otro lado, se lograron muy buenos resultados de clasificación utilizando el clasificador SVM-C con parámetros optimizados mediante girdsearch $C = 10$ y $\gamma = 100$. Los resultados indicaron que el problema de clasificación de bacterias en estas cuatro clases, requiere de fronteras de decisión de alta complejidad, logrando como mejor resultado una accuracy 96,1% con un clasificador con 4086 vectores de soporte.

A partir de la utilización de PCA, y del método de extracción de features desarrollado, se entrenó un nuevo clasificador SVM-C con los parámetros optimizados sobre un nuevo espacio de features reducido mediante la selección y extracción de features. Los resultados alcanzados muestran que la selección de features puede resultar beneficiosa al descartar las features que no agregan información pertinente para la tarea de clasificación entre los genomas.

Como trabajo a futuro queda la incógnita de hasta qué punto aumentar el largo de los n-gramas puede ser beneficioso para la clasificación mediante SVM-C. Además, dada la complejidad del modelo necesaria para clasificar el conjunto de datos, queda pendiente investigar si otros modelos que ofrecen mayores niveles de complejidad, como pueden ser las redes neuronales, pueden lograr mejores resultados.

References

[1] John C. Wooley, Adam Godzik, and Iddo Friedberg. A primer on metagenomics. *PLOS Computational Biology*, 6(2):1–13, 02 2010.

[2] Elaine Garbarine, Joseph DePasquale, Vinay Gadia, Robi Polikar, and Gail Rosen. Information-theoretic approaches to svm feature selection for metagenome read classification. *Computational Biology and Chemistry*, 35(3):199 – 209, 2011.

[3] Miten Jain, Hugh E. Olsen, Benedict Paten, and Mark Akeson. The oxford nanopore minion: delivery of nanopore sequencing to the genomics community. *Genome Biology*, 17(1):239, Nov 2016.