Predicción de interacción entre péptido y el complejo mayor de histocompatibilidad tipo I con Artificial Neural Networks (ANN) y Support Vector Machines (SVM)

Machine Learning - PEC 2

Jorge Vallejo Ortega

25 de mayo, 2020

Sumario

1 Algoritmo Red Neuronal Artificial

Los algoritmos de Red Neuronal Artificial (o *ANN* por el inglés *'Artificial Neuronal Net'*), son simulaciones que imitan el funcionamiento de las redes neuronales biológicas. Por lo general no modelan redes neuronales concretas, sino modelos abstractos.

Como abstracción de una red neural biológica, se podría decir que las ANNs constan de dos tipos de "piezas": **nodos** y **vértices**.

- Los **nodos** de las ANNs son objetos de código que - igual que hacen las neuronas biológicas - aceptan datos, los procesan, y envían datos a su vez.
- Los **vértices** son el equivalente a los axones y dendritas. Definen qué neuronas están conectadas entre ellas y, por tanto, cómo viaja la información dentro de la red neuronal.

Cada ANN se puede describir por las siguientes características:
- La **función de activación**.
- La **topología**.
- El **algoritmo de entrenamiento**.

La **función de activación** describe la forma en que los nodos combinan los datos que les llegan para generar los datos que envían. Diferentes funciones de activación son más adecuadas para diferentes tareas de aprendizaje. Las funciones más comunes son la *función de paso único*, la *lineal*, la *lineal saturada*, la *tangente hiperbólica* y la *gaussiana*.

La **topología** de la red describe el número de nodos por capa, el número de capas, y cómo se conectan unas con otras (número de nodos y si la información puede viajar "hacia atrás", de capas posteriores a capas anteriores).

El **algoritmo de entrenamiento** configura la forma en que se ponderan los datos que se transmiten por cada vértice, una forma de modular las señales que viajan por la red *adicional a la función de activación*.

1.1 Fortalezas y debilidades de las Redes Neuronales Artificiales

Fortalezas	Debilidades
- Se pueden usar tanto para problemas de clasificación como para predicción numérica.	- Necesitan enormes cantidades de recursos de cálculo y su entrenamiento es lento, especialmente cuando la topología de la red es compleja
- Pueden modelar patrones más complejos que casi cualquier otro algoritmo.	- Muy susceptibles a sobreajustar los datos de entrenamiento.
- Hace pocas asunciones previas acerca de las relaciones entre los datos.	- El modelo resultante del entrenamiento es de tipo caja negra, complejo y difícil (si no imposible) de interpretar.

2 Algoritmo Support Vector Machine

Los algoritmos SVM (*Support Vector Machine*) procesan datos distribuidos en un espacio multidimensional, y los utilizan para definir una superficie (hiperplano) que divide dicho espacio en regiones homogéneas que agrupan observaciones afines. Estos algoritmos se usan en el campo del aprendizaje automático tanto para clasificar observaciones, como para hacer predicciones. Este pequeño script pondrá en aplicación una de sus funciones más sencillas, la clasificación de un conjunto de observaciones en dos grupos diferenciados.

Cuando las observaciones de ambos grupos están claramente separadas y pueden separarse completamente mediante una línea, plano, o su equivalente multidimensional (hiperplano), hablamos de **datos separables**

linealmente. Sin embargo, en la vida real es muy común que la relación entre variables no sea lineal. Es estos casos, se pueden seguir dos estrategias: aplicación de la *slack variable* (variable fluctuante) y el uso del *kernel trick* (cambio de la función kernel).

El caso de la variable *slack* consiste en añadir una variable el algoritmo que "permite" dejar observaciones de entrenamiento fuera del grupo al que propiamente pertenecen. Aquí lo que produce el algoritmo es una optimización del hiperplano para miniminar la distancia de esas observaciones hasta el límite que define su clasificación correcta.

El *kernel trick* es matemáticamente más complejo y consiste en añadir nuevas dimensiones a los datos y usarlas para identificar las relaciones no lineales entre las variables. Aquí las superficies que definen los límites entre grupos ya no son hiperplanos, sino curvas de mayor o menor complejidad.

2.1 Fortalezas y debilidades del algoritmo SVM

Fortalezas	Debilidades
- Se puede usar tanto para problemas de clasificación como para predicción numérica.	- Encontrar el modelo más adecuado requiere probar varias combinaciones de kernels y parámetros.
- Poco sensible al ruido en los datos y poco propenso al sobreajuste (overfitting).	- El entrenamiento puede ser lento, en especial se el set de datos tiene un gran número de características o ejemplos.
- Puede ser de más fácil uso que las redes neurales, especialmente debido a la existencia de varios algoritmos SVM que han sido adaptados para su uso en ciencia de datos.	- El modelo resultante del entrenamiento es de tipo caja negra, complejo y difícil (si no imposible) de interpretar.
- Popular debido a su alta precisión y a la fama debida a su uso ganador en competiciones de minería de datos.	

3 Análisis exploratorio

El set de datos para este informe proviene del fichero "peptidos.csv".

El dataset está compuesto por:
15840 observaciones, de cada una de las cuales se han obtenido
2 variables.

```
## 'data.frame':    15840 obs. of  2 variables:
##  $ sequence: chr  "LMAFYLYEV" "HQRLAPTMP" "FMNGHTHIA" "WLLIFHHCP" ...
##  $ label   : Factor w/ 2 levels "NB","SB": 2 1 2 1 1 2 2 1 1 1 ...
```

La primera variable, 'sequence', corresponde a la secuencia de aminoácidos en cada péptido en la que cada aminoácido está codificado por una letra siguiendo las recomendaciones de la IUPAC(IUPAC 1984).

La segunda variable, 'label', nos informa de la interacción del péptido con el MHCI; donde 'NB' significa que no hay interacción, y 'SB' significa que sí hay interacción.

Un dato que nos interesaría conocer es el patrón que sigue la secuencia de péptidos en cada categoría ('NB' y 'SB'), representarlo como secuencia logo, y ver si hay alguna diferencia apreciable a simple vista.

Antes de nada, comprobamos si tenemos secuencias repetidas en nuestros datos:
- Número de observaciones: **15 840**.
- Número de secuencias únicas: **15 840**.

Y la longitud de secuencia de todos los péptidos:

Longitud	Péptidos
9	15 840

- Sequencias sin interacción con el MHCI: **7 920**
- Sequencias que interaccionan con el MHCI: **7 920**

Una vez aclarada la estructura de las secuencias pasamos a generar los logos de secuencia; en este caso usando el paquete `ggseqlogo`(Wagih 2017).

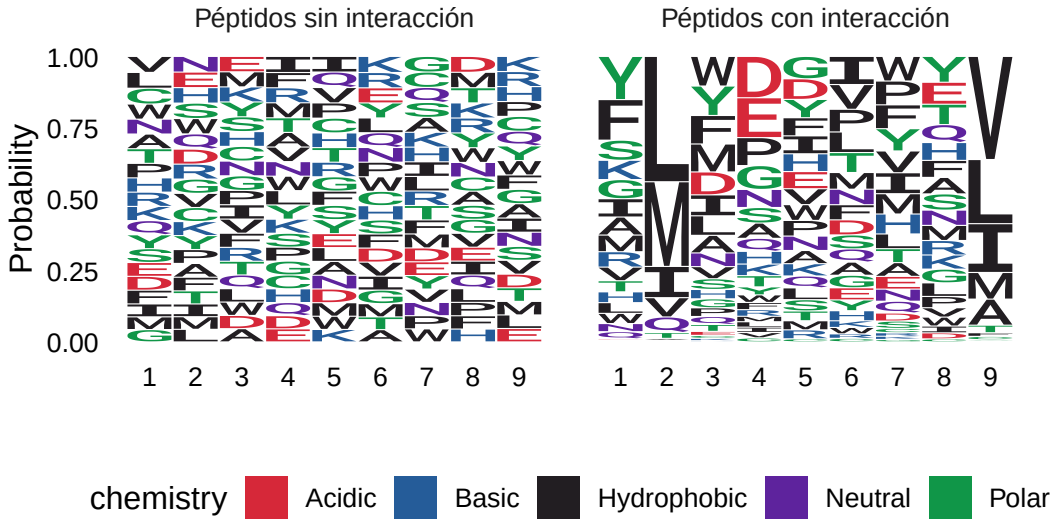

Figure 1: Logo de secuencia de péptidos sin interacción con MHCI (superior) y con interacción (inferior). En el eje vertical se representan las proporciones de cada aminoácido en cada posición.

Al comparar los logos de secuencia de los péptidos que no muestran interacción con el MHCI (superior), y los que sí interactúan (inferior); llama la atención la prevalencia de aminoácidos hidrofóbicos en las posiciones 2 y 9 de los péptidos con interacción.

La diferencia se ve mucho mejor si en el eje vertical representamos los bits de información aportados en lugar de la probabilidad:

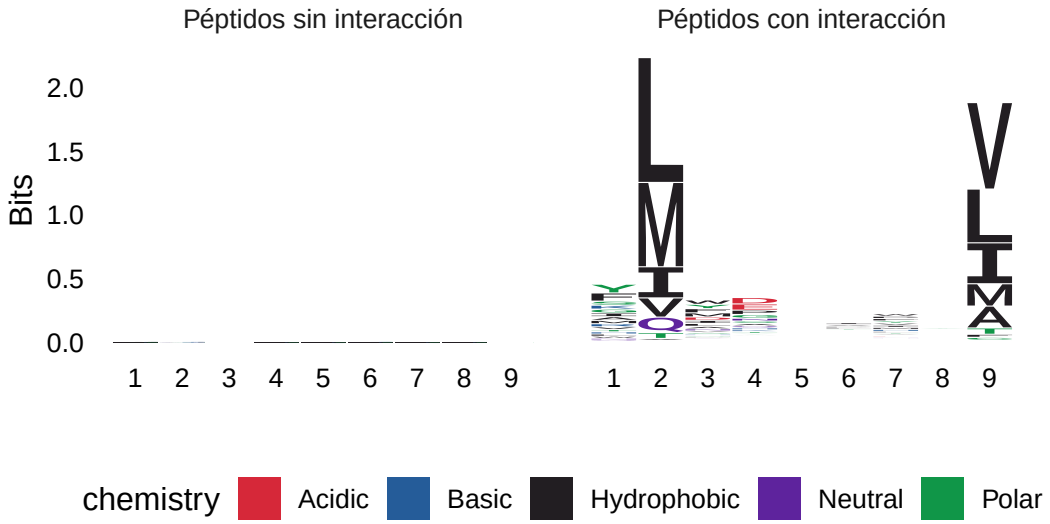

Figure 2: Logo de secuencia de péptidos sin interacción con MHCI (superior) y con interacción (inferior). En el eje vertical se representan los bits de información de cada aminoácido en cada posición.

En el caso de los péptidos sin interacción con MHCI, la información aportada por cada aminoácido es tan baja que apenas aparecen representados cuando se usa la misma escala que para los péptidos que sí interaccionan.

4 Pre-procesado de datos

4.1 One-shot encoding

Como paso previo al análisis mediante aprendizaje automático, los péptidos han sido recodificados mediante la transformación conocida como *one hot encoding*, según la cual cada aminoácido en la secuencia del péptido es representado en código binario.

Hecho esto, vemos que los datos con los que alimentaremos los algoritmos de aprendizaje automático se componen de **15 840** secuencias y **181** variables cada secuencia.

La siguiente secuencia es un ejemplo de péptido (LMAFYLYEV) codificado mediante one-hot encoding:

```
##   [1] 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
##  [38] 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
##  [75] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1
## [112] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0
## [149] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
```

4.2 Sets de entrenamiento y prueba

Como último paso antes del análisis, hemos dividido el set de datos - al azar - en un subset de entrenamiento (2/3 de las observaciones), y un subset de evaluación (1/3 de las observaciones).

Entrenamiento: 10 560 observaciones.
Evaluación: 5 280 observaciones.

5 Análisis de los datos con Red Neuronal Artificial

5.1 Primer modelo (una capa oculta con un nodo)

El primer modelo consta de:
180 nodos de input
1 capa oculta compuesta por 1 nodo
2 nodos output

Error: 1.0148474
Pasos de entrenamiento: 147

5.2 Segundo modelo (una capa oculta con tres nodos)

El segundo modelo consta de:
180 nodos de input
1 capa escondida compuesta por 3 nodos
2 nodos output

Error: 1.0233313
Pasos de entrenamiento: 89

5.3 Evaluación de los modelos

Para evaluar los modelos utilizamos el set de datos de prueba que habíamos especificado con anterioridad.

A la hora de evaluar modelos predictivos hay tres parámetros sencillos de calcular y explicar, y son los que se han usado en este informe:

- **Sensibilidad**: Es la proporción de observaciones positivas que han sido clasificadas correctamente.

$$\text{sensibilidad} = \frac{\text{positivos auténticos}}{\text{positivos auténticos } + \text{ falsos negativos}}$$

- **Especificidad**: Es la proporción de observaciones negativas que han sido clasificadas correctamente.

$$\text{especificidad} = \frac{\text{negativos auténticos}}{\text{negativos auténticos } + \text{ falsos positivos}}$$

- **Precisión**: Es la proporción de ejemplos clasificados como positivos que son *realmente* positivos.

$$\text{precisión} = \frac{\text{positivos auténticos}}{\text{positivos auténticos } + \text{ falsos positivos}}$$

El objetivo del modelo de clasificación que hemos construido es distinguir entre péptidos que interactúan con el MHCI (positivo) y péptidos que no interactúan con el MHCI (negativo).

En la siguiente tabla podemos comparar los tres parámetros de evaluación para ambos modelos (con 1 nodo o con 3 nodos en la capa oculta):

	Sensibilidad	Especificidad	Precisión
Un nodo oculto	0.9996	0.9985	0.9985
Tres nodos ocultos	0.9992	0.9989	0.9989

Ambas redes obtienen muy buenos resultados, sin embargo **la red con un único nodo en la capa oculta es ligeramente mejor en cuanto a sensibilidad**; mientras que el modelo construido con la red de **tres**

nodos ocultos es ligeramente mejor en los valores de evaluación de **especificidad** y **precisión**.

5.4 Modelo de una capa oculta con tres nodos entrenado por validación cruzada de 5 particiones

El método de *validación cruzada por particiones* que se usa para evitar entrenar o testear los modelos con datos no representativos, y ofrece unos resultados de evaluación del modelo más fiables respecto a cómo se comportará al enfrentarse a datos nuevos.

	Sensibilidad	Especificidad	Precisión
Un nodo oculto	0.9996	0.9985	0.9985
Tres nodos ocultos	0.9992	0.9989	0.9989
Validación cruzada	0.9962	0.9994	0.9978

Si comparamos los datos de **sensibilidad, especificidad y precisión**, con los calculados anteriorment para el modelo de tres nodos ocultos, **los nuevos valores son inferiores** en los tres casos. Aunque es el mismo modelo; al estar calculados usando diferentes combinaciones del mismo dataset representan mejor las capacidades del modelo.

6 Análisis de datos con Support Vector Machine

6.1 Entrenamiento y evaluación de modelos

Para este informe hemos construido y entrenado dos modelos SVM de diferente complejidad para comparar sus resultados.

Uno es un **SVM lineal**, más sencillo, y el otro es un **SVM gaussiano**; este último es más complejo y requiere más recursos de cálculo, pero permite que el modelo aprenda relaciones más complejas entre las variables del set de datos.

Los sets de entrenamiento y evaluación usados para ambos modelos son idénticos, para poder hacer la comparación, e iguales también a los usados en los modelos de Red Neuronal Artificial.

A la hora de evaluar modelos predictivos usamos los mismos tres parámetros usados anteriormente con los modelos de Red Neuronal Artificial: *sensibilidad*, *especificidad* y *precisión*. Podemos ver la comparativa en la siguiente tabla:

	Sensibilidad	Especificidad	Precisión
Modelo lineal	0.999	0.998	0.998
Modelo Gaussiano	1.000	0.990	0.990

Vemos en la comparativa que **el modelo gaussiano es ligeramente más sensible** que el lineal (menos falsos negativos), pero a cambio de **menor especificidad y precisión** (más falsos positivos).

La elección entre uno y otro modelo en este caso podría depender si no queremos pasar por alto ninguno de los péptidos que presentan interacción (elegiríamos el modelo gaussiano, con mayor sensibilidad); o de si no queremos asumir la pérdida de tiempo y dinero que supondría testar falsos positivos (eligiríamos el modelo lineal, con mayor especificidad y precisión).

6.2 Modelo gaussiano entrenado por validación cruzada de 5 particiones

	Sensibilidad	Especificidad	Precisión
Modelo Lineal	0.999	0.998	0.998
Modelo Gaussiano	1.000	0.990	0.990
Gaussiano por validación cruzada	0.999	1.000	0.999

El modelo gaussiano por validación cruzada es ligeramente menos sensible que el que no está entrenado por validación cruzada, pero su especificidad y precisión son mayores.Y en todos los valores presenta mejor desempeño que el modelo lineal.

Además los datos de desempeño en modelos entrenados por validación cruzada son más robustos en cuanto a cómo se comportará el modelo frente a datos desconocidos. Por todas estas razónes, **el modelo más adecuado de SVM sería el gaussiano entrenado por validación cruzada**.

7 Discusión final

Finalmente comparamos el desempeño de todos los modelos, tanto los producidos mediante Redes Neurales Artificiales como los producidos mediante Support Vector Machines, para elegir el que mejor se adapte a nuestras necesidades:

	Sensibilidad	Especificidad	Precisión
Un nodo oculto	0.9996	0.9985	0.9985
Tres nodos ocultos	0.9992	0.9989	0.9989
Validación cruzada	0.9962	0.9994	0.9978
Modelo Lineal	0.9989	0.9981	0.9981
Modelo Gaussiano	1.0000	0.9898	0.9898
Gaussiano por validación cruzada	0.9989	0.9996	0.9992

Vemos que no hemos conseguido una solución perfecta, y además no hay grandes diferencias en los valores de desempeño. La elección del modelo a elegir depederá de si preferimos **sensibilidad**, para lo cual elegiríamos el **modelo SVM gaussiano**; o si necesitamos que **especificidad y precisión** sean lo más altas posible, por lo que nos decantaríamos por el **modelo SVM gaussiano entrenado por validación cruzada**.

Aunque también debemos tener en cuenta que los datos de desempeño de este último modelo, el **SVM gaussiano entrenado por validación cruzada**, se han obtenido al entrenarlo con una mayor variedad de datos que otros modelos SVM, por lo que es un desempeño más representativo de cómo se comportará el modelo al enfrentarlo a datos nuevos; así que podría ser la mejor elección en cualquier caso.

8 Apéndice A: Código

El documento original en formato .Rmd, que incluye el código completo en lenguaje R usado para generar este informe, se puede consultar y descargar en el siguiente repositorio de Github: jorgevallejo/interaccion_peptido_MHCI

9 Apéndice B: Reproducibilidad

```
sessionInfo() # For better reproducibility
```

```
## R version 3.6.3 (2020-02-29)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 18363)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=Spanish_Spain.1252  LC_CTYPE=Spanish_Spain.1252
## [3] LC_MONETARY=Spanish_Spain.1252 LC_NUMERIC=C
## [5] LC_TIME=Spanish_Spain.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] MLmetrics_1.1.1  kernlab_0.9-29   caret_6.0-86     lattice_0.20-38
##  [5] RSNNS_0.4-12     Rcpp_1.0.4.6     neuralnet_1.44.2 ggseqlogo_0.1
##  [9] ggplot2_3.3.0    knitr_1.28
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_1.0.0     xfun_0.13            reshape2_1.4.4
##  [4] purrr_0.3.4          splines_3.6.3        colorspace_1.4-1
##  [7] vctrs_0.2.4          generics_0.0.2       stats4_3.6.3
## [10] htmltools_0.4.0      yaml_2.2.1           survival_3.1-8
## [13] prodlim_2019.11.13   rlang_0.4.5          e1071_1.7-3
## [16] ModelMetrics_1.2.2.2 pillar_1.4.3         glue_1.4.0
## [19] withr_2.2.0          foreach_1.5.0        lifecycle_0.2.0
## [22] plyr_1.8.6           lava_1.6.7           stringr_1.4.0
## [25] timeDate_3043.102    munsell_0.5.0        gtable_0.3.0
## [28] recipes_0.1.12       codetools_0.2-16     evaluate_0.14
## [31] labeling_0.3         class_7.3-15         highr_0.8
## [34] scales_1.1.0         ipred_0.9-9          farver_2.0.3
## [37] digest_0.6.25        stringi_1.4.6        dplyr_0.8.5
## [40] grid_3.6.3           tools_3.6.3          magrittr_1.5
## [43] tibble_3.0.1         crayon_1.3.4         pkgconfig_2.0.3
## [46] ellipsis_0.3.0       MASS_7.3-51.5        Matrix_1.2-18
## [49] data.table_1.12.8    pROC_1.16.2          lubridate_1.7.8
## [52] gower_0.2.1          assertthat_0.2.1     rmarkdown_2.1
## [55] iterators_1.0.12     R6_2.4.1             rpart_4.1-15
## [58] nnet_7.3-12          nlme_3.1-144         compiler_3.6.3
```

Referencias

IUPAC, IUB. 1984. “IUPAC-Iub Joint Commission on Biochemical Nomenclature (Jcbn). Nomenclature and Symbolism for Amino Acids and Peptides. Recommendations 1983.” *Biochem J* 219 (2): 345–73.

Wagih, Omar. 2017. “ggseqlogo: a versatile R package for drawing sequence logos.” *Bioinformatics* 33 (22): 3645–7. https://doi.org/10.1093/bioinformatics/btx469.