Desarrollar un filtro de detección de Spam usando redes neuronales artificiales (ANN)

Diego González Rodríguez

21 de marzo de 2015

Abstract
Existen diversos enfoques para la detección de spam que es enviado a lo largo y ancho de la red, desde enfoques legislativos hasta aquellos que implican un aprendizaje estadístico, que es en el que nos vamos a centrar en este trabajo. Concretamente, la técnica empleada consiste en analizar el contenido de un mensaje de correo electrónico para determinar si es spam o no. Para llevar a cabo esta solución nos hemos basado en el trabajo previamente realizado por [1], donde haciendo uso de una red neuronal artificial (ANN) se determina la naturaleza de un mensaje de correo.

Introducción.

El presente documento sirve como memoria para el trabajo desarrollado para la asignatura ***Redes Neuronales y aprendizaje estadistico*** del master ***Tratamiento estadístico-computacional de la información***. Dicho documento se ha dividido de la siguiente forma:

- ***Impacto del Spam***: donde brevemente explicaremos porque hemos seleccionado este tema como práctica final de la asignatura.
- ***Solución planteada***: trataremos de dejar claro tanto el enfoque como la solución planteada en el ejercicio.
- ***Funcionamiento***: por si el lector estuviera interesado en realizar sus propias pruebas explicaremos como funciona la aplicación desarrollada.
- ***Resultados obtenidos***: detallaremos el experimento realizado para evaluar la solución desarrollada.
- ***Conclusiones y trabajo futuro***: comentaremos dificultades encontradas en la implementación del sistema, así como posibles mejoras si se quisiera profundizar más en el problema.

Impacto del Spam.

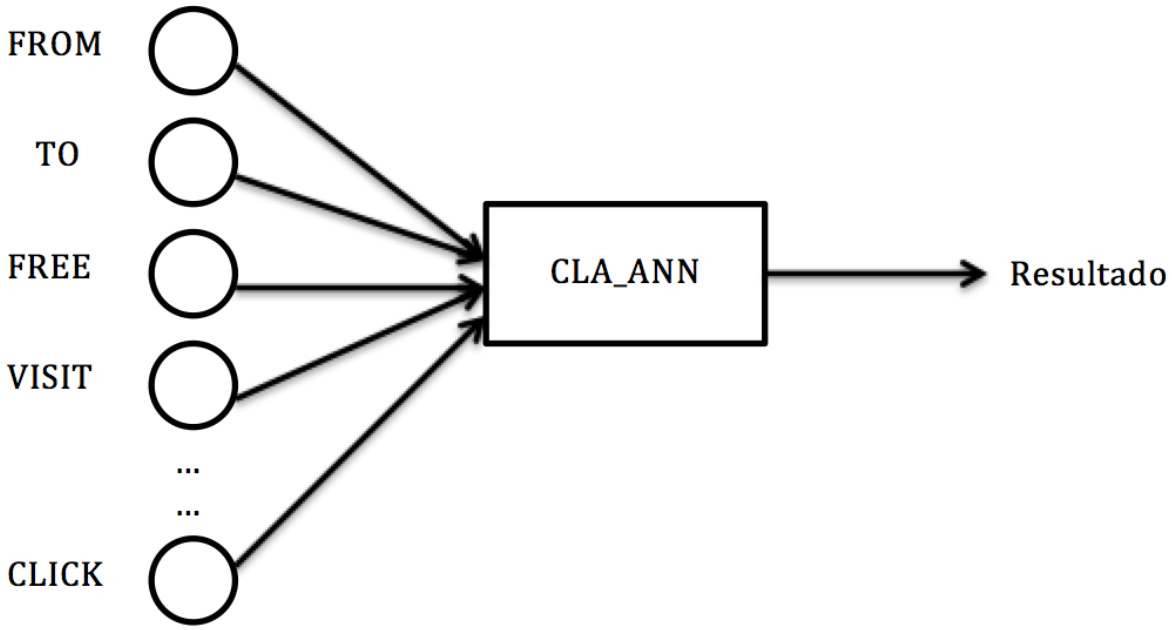
El correo electrónico es una forma eficiente de comunicación que se ha adoptado globalmente tanto en indiviudos como en organizaciones. Hoy en día, más y más gente confía en dicho sistema para entablar comunicaciones entre amigos, familia, compañeros, clientes y asociados. Desgraciadamente, dicho extensión en su uso continuo lo ha llevado a ser considerado como una fuente de entrada a ciertas amenazas, en particular el conocido como spam o correo no solicitado se ha convertido en una amenaza dificil de detectar, la cual es recibida en un alto volumen. Según [2] en octubre del año 2006 el 67% del tráfico de correos electrónicos se correspondía con spam, subiendo del 57% del año anterior. El correo no deseado se ha convertido en un serio problema para los servicios de correo, de forma que es una dificil tarea encontrarse correos deseados en la carpeta de recibidos de nuestros sistemas. Hay que tener en cuenta que la existencia de spam es un problema económico para las empresas, ya que para las compañias provedoras de Internet supone una perdida en el ancho de banda, y para las organizaciones supone una perdida en la productividad de sus trabajadores. Es

por esto que se ha convertido en un problema en auge que requiere la aplicación del aprendizaje estadístico, entre otras soluciones.

Solución planteada.

La lucha por convertirse en el dominador en el servicio de correos electrónicos por parte de las mayores y mejores compañias de Internet, como Google, Yahoo o Microsoft, provocó que la detección de correos no deseados fuera un tema al que se le ha dedicado muchas horas de investigación. Basicamente y a grandes rasgos existen dos disciplinas, basadas en filtrar la dirección IP de origen y en filtrar el contenido del mensaje. La primera solución es muy solida pero la forma de saltar dicha defensa es muy sencilla, cosa que no ocurre con la segunda formula. Además, unido a la naturaleza de la asignatura impartida hemos escogido una solución basada en analizar el contenido del mensaje usando el entrenamiento del perceptron.

Graficamente el esquema que hemos seguido para resolver el problema del filtrado de spam es el siguiente:



donde cada neurona se correspondería con una palabra frecuente contenida en el mensaje a analizar, de forma que una vez fijadas las palabras a buscar en futuros mensajes, a tráves de una fase de entrenamiento, y sus pesos, mediante una simple formula que tiene en cuenta la cantidad de veces que aparece dicha palabra en un mensaje deseado y en uno considerado como spam, se podría obtener el valor total del peso en un mesaje, esto es la suma de los pesos de aquellas palabras que aparezcan en el mensaje a analizar. Con lo que si dicho peso final superara cierta umbral, establecido por el usuario, se determinaría si es considerado como mensaje deseado o como spam.

La implementación de dicha solución se ha llevado a cabo en el lenguaje *R*, donde distinguimos cuatro módulos que pasamos a comentar.

Programa principal.

El encargado de ejecutar el programa de filtrado de spam. Se puede encontrar en el fichero ***main.R***. Se correspondería con el siguiente pseudocódigo.

```
Start training
For each message do
 Start cla_ann
 Start learning
End
Calculate statistics
```

En este modulo se debe establecer el número de neuronas de nuestra red (***num_node***), así como establecer una ruta válida que contenga los mensajes a analizar (***validation_set_directory***). Como se pretende analizar la fiabilidad del sistema es necesario establecer la naturaleza de cada correo, primera columna del data frame res, para poder ser comparado con el resultado de la evaluación.

Entrenamiento.

Se encarga de entrenar la red neuronal, estableciendo las palabras más frecuentes que se pueden encontrar en un mensaje de spam, así como el peso de cada neurona. Se puede encontrar en el fichero ***training.R***. Se correspondería con el siguiente pseudocódigo.

```
For each token in the spam message corpus do
 If layer is already exist in Innate_Input_Layer then
    Innate_Input_Layer.msg_matched->Innate_Input_Layer.msg_matched + 1
    Innate_Input_Layer.spam_matched->Innate_Input_Layer.spam_matched + spam_increment
 else
  Add token to Innate_Input_Layer
  Innate_Input_Layer.msg_matched->Innate_Input_Layer.msg_matched + 1
  Innate_Input_Layer.spam_matched->Innate_Input_Layer.spam_matched + spam_increment
 end if
end for
```

Para poder entrenar el perceptron se debe proporcionar dos directorios que contenga mensajes de spam (***training_spam_set_directory***) y mensajes deseados (***training_ham_set_directory***).

CLA_ANN.

El módulo que determina si un mensaje es spam o deseado en función de la suma del peso de aquellas palabras frecuentes, previamente establecidas en la fase de entrenamiento. Se puede encontrar en el fichero ***cla_ann.R***. Se correspondería con el siguiente pseudocódigo.

```
For each token in Innate_Input_Layer do
 if token matches message then
   total_weight <- total_weight + token.weight
   number_of_token_matched= number_of_token_matched+1
   if token.weight > threshold then
     Desired_Output <- 0.9999 (Spam)
   else
     Desired_Output <- 0.1111 (Ham)
   end if
 end if
end for
score <- total_weight / number_of_token_matched
if score > threshold then
```

```
 Message is spam
 error_rate <- Absolute (Desired_Output - Score)
 correction <- Error_rate*Learning_rate
 Token.weight <- Token.weight + Correction
 If token does not exist in Innate _ Input_Layer then
   Add token to Adaptive_ Input_Layer (This is to represent continuous learning)
 end if
else
 Message is not spam
 error_rate <- Absolute (Desired_Output - Score)
 correction <- Error_rate*Learning_rate
 Token.weight <- Token.weight - Correction
end if
```

Para determinar la naturaleza del mensaje se debe establecer un umbral (***threshold***).

Aprendizaje.

Para mejorar los resultados obtenidos incorporamos un módulo de aprendizaje automático. Se puede encontrar en el fichero ***cla_ann.R***. Se correspondería con el siguiente pseudocódigo.

```
if criteria happened then
  Merge Adaptive_ Input_Layers with Innate_Input_Layers and then order it descending based on Token.spa
end if
Select top Innate_Input_Layers
Adaptive_ Input_Layers <- Empty
```

Funcionamiento.

Para poder ejecutar el programa solo se necesita establecer una serie de directorios que contengan mensajes de spam y deseados tanto para entrenar al perceptron como evaluar su rendimiento. No obstante, se proporciona una serie de mensajes de correo electrónico, que han sido obtenidos en [3], para su posible ejecución. Dichos mensajes están estructurados en los siguientes directorios:

- ***training_ham_set:*** conjunto de correos deseados para la fase de entrenamiento.
- ***training_spam_set:*** conjunto de correos spam para la fase de entrenamiento.
- ***validation_ham_set:*** conjunto de correos deseados para la fase de evaluación.
- ***validation_spam_set:*** conjunto de correos spam para la fase de evaluación.

Por último, se deberá esteblecer un umbral que determinará si un mensaje es deseado o spam, así como el número de nodos de nuestra red, dicho de otra forma, el número de palabras frecuentes a buscar en cada correo. Dichas variables establecerán el funcionamiento final del sistema como veremos en el siguiente punto.

Una vez introducidos los parametros de configuración bastará con ejecutar el script ***main***. Es necesario recordar que tanto el entrenamiento como la evaluación de los mensajes llevará tiempo, en función de la cantidad de mensajes usadas será mayor o menor.

NOTA: el código se corresponde a la tercera iteración realizada, que como veremos se corresponde con el perceptron con aprendizaje continuo donde se elimina aquellas palabras con valor 1 en los campos ***msg_matched*** y ***spam_matched***.

Resultados obtenidos.

La evaluación de la solución propuesta se ha realizado durante tres iteraciones donde ibamos integrando mejoras en la implementación de la solución. Para cada iteración se ha usado el siguiente conjunto de datos.

- 1651 mensajes deseados para la fase de entrenamiento.
- 1398 mensajes de spam para la fadse de entrenamiento.
- 2501 mensajes deseados para la evaluación del perceptron.
- 501 mensajes de spam para la evaluación del perceptron.

Para cada una de las distintas iteraciones se han probado distintas configuraciones del perceptron en busca del mejor valor, tanto como para la detección de spam como de correo deseado. Pasamos a detallar cada una a continuación.

1ª Iteración.

En primer lugar implementamos el filtro anti-spam haciendo uso del perceptron, tal como es explicado en [1] lo más simple posible. Esto consistiría en los modulos de entrenamiento y el propio algoritmo ann. Dando el siguiente resultado.

Numero de neuronas (nodos)	Threshold	SPAM	NO SPAM
100	0.8	74.42	41.24
500	0.9	0.63	100
500	0.8	25.37	96.12
500	0.6	73.78	15.28
1000	0.8	94.03	14.55
1000	0.75	35.02	94.18
1000	0.7	81.15	50.63
1000	0.6	45.91	73.41

Donde observamos que la mejor combinación sería un perceptron de **1000 neuronas** y un **umbral de 0.7**, pero podemos observar como los resultados son muy dispares en ciertas configuraciones.

2ª Iteración.

En [1] mencionan que las entradas con valor 1 en las columnas **msg_matched** y **spam_matched** son inútiles a la hora de entrenar el perceptron, de modo que añadimos una función (**delete_insignificant_tokens**) que se encargue de eliminar dichas entradas de **innate_input_layers**. Dando los siguientes resultados.

Numero de neuronas (nodos)	Threshold	SPAM	NO SPAM
100	0.8	39.69	85.28
500	0.9	0.63	100

Numero de neuronas (nodos)	Threshold	SPAM	NO SPAM
500	0.8	94.84	26.22
500	0.6	72.94	15.02
1000	0.8	21.73	98.06
1000	0.75	42.41	81.82
1000	0.7	58.38	58.38
1000	0.6	75.11	18.08

En este caso no hay diferencias entre la solución con **100 neuronas** y un **umbral de 0.8** y la de **1000 neuronas** y **umbral de 0.75**, de modo que podríamos seleccionar la primera de forma que los recursos que consume son menores y la diferencia no es significativa para ese mayor consumo. No obstante, no hay mejora significativa respecto de la iteración 1.

3ª Iteración.

No hemos conseguido implementar de forma satisfactoria el módulo de aprendizaje continuo explicada en [1], de forma que nuestros resultados no han podido mejorar en ningún momento lo alcanzado previamente. Nos hemos encontrado con varios problemas referentes a la actualización de los pesos dando como resultado valores mayores que 1, de modo que solo era capaz de detectar correo deseado.

Conclusiones y trabajo futuro.

Antes de ver las conclusiones obtenidas vamos a destacar brevemente las dificultades encontradas:

- El autor de este trabajo no tiene experiencia previa en proyectos de envergadura desarrollados en R, de forma que el tratamiento del texto y la estructuración del código han sido una parte difícil en el desarrollo del conjunto presentado aquí. Sin saber además si la solución planteada sería la óptima.

- En [1] presentan el algoritmo usado y dan una breve explicación de como funciona, pero nuestra aproximación ha sido más académica de forma que hay simplificaciones que hemos aplicado que han podido llevar a un resultado distinto a [1].

- El tiempo ha sido un factor decisivo a la hora de no poder continuar perfeccionando el algoritmo, ya que debido a la existencia de otras prácticas y al hecho de que el autor trabaja se ha decidido terminar la investigación en este punto.

Indudablemente los resultados comparados con [1] son muy inferiores, no obstante, hemos desarrollado e implementado un mecanismo de aprendizaje estadístico en forma de perceptron, que era el objetivo de la práctica, con el fin de ser capaces de detectar correo spam. Además hemos podido comprobar la importancia de manejar el peso de los nodos de forma precisa hasta el punto de que en casos donde el número de palabras frecuentes disponibles era mucho menor se ha conseguido mejorar el porcentaje de detección de spam a otros que requerían un computo mayor. Con lo que podemos estar satisfechos con el trabajo realizado.

Indudablemente, en la opinión del autor hay dos frentes a mejorar en el futuro en este trabajo.

- El tratamiento y detección de palabras contenidas en un correo debería mejorarse de forma que solo nos centraramos en el propio contenido y no en palabras que pertenecen al formato del propio correo, ya

que a pesar de que hemos desestimado ciertas palabras al final hemos descubierto palabras frecuentes que lo erán porque son palabras reservadas en el formato usado en un correo.

- El aprendizaje continuo debe ser revisado para poder estar cerca de los valores que se han alcanzado en [1].

Bibliografía.

[1] A. T. Sabri, A. H. Mohammads, B. Al-Shargabi, M. A. Hamdeh. Developing New Continuous Learning Approach for Spam Detection using Artificial Neural Network (CLA_ANN). European Journal of Scientific Research ISSN 1450-216X Vol.42 No.3 (2010), pp.511-521.
[2] MessageLabs. Threat Statistics. December 2006, http://www.messagelabs.com/Threat_Watch/Threat_Statistics/.
[3] Corpus encontrado en Spam Assasin: https://spamassassin.apache.org/publiccorpus/
[4] Referencia: http://stackoverflow.com/