PER (E.T.S. de Ingeniería Informática)
Curso 2018-2019

Práctica 2. Clasificación de textos

Jorge Civera Saiz, Carlos D. Martínez Hinarejos
Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València

Índice

1. Objetivos

- Desarrollar un clasificador de textos, más concretamente un filtro *anti-spam*, basado en el clasificador multinomial estudiado en clase de teoría.

- Resolver los problemas prácticos que nos podemos encontrar durante el desarrollo de un clasificador de textos (representación en memoria, suavizado de parámetros, etc.).
- Diseñar un conjunto de experimentos para evaluar el clasificador de textos implementado.
- Analizar el comportamiento (evolución del error) del clasificador implementado en función de sus parámetros.

2. El corpus TREC 2006

El corpus TREC 2006[1] es una tarea de clasificación de correos electrónicos en *ham*[2] y *spam* que se publicó en la *Text REtrieval Conference (TREC)* en el *Spam Track*. En realidad esta tarea está compuesta por dos subtareas, la versión en inglés (*trec06p*) y la versión en chino (*trec06c*). En esta práctica nos centraremos en *trec06p*.

El corpus *trec06p* dispone de 37822 correos electrónicos clasificados mediante expertos humanos asistidos por software anti-spam en dos clases, *ham* (12910) y *spam* (24912).

Dado que es una tarea real, algunos correos de *spam* pueden contener todo tipo de software virulento, por lo que te recomendamos que **NO abras estos correos con un cliente de correo o navegador web**.

Una versión simplificada del corpus TREC 2006 está disponible en el directorio de la práctica `/labos/asignaturas/ETSINF/per/p2/dat/` incluyendo:

- Fichero `index` con la lista de ficheros (rutas) y la etiqueta de clase (*spam* o *ham*) de cada fichero.
- Directorio `data` con los correos electrónicos en ficheros individuales y estructurados en directorios.

Las primeras líneas del fichero `index` tienen el siguiente aspecto:

```
$ less index
ham data/000/000
spam data/000/001
spam data/000/002
ham data/000/003
...
```

La primera columna indica la etiqueta de clase del fichero que se encuentra en la ruta indicada por la segunda columna.

El directorio `data` está organizado en subdirectorios enumerados que contienen los 37822 correos electrónicos.

[1] `http://plg.uwaterloo.ca/~gvcormac/trecspamtrack06/`

[2] Correo deseado.

```
data/
|-- 000
|   |-- 000
|   |-- 001
...
|   |-- 298
|   `-- 299
|-- 001
|   |-- 000
|   |-- 001
...
|   |-- 298
|   `-- 299
`-- 126
    |-- 000
    |-- 001
    ...
```

Es posible visualizar el contenido de un correo electrónico.

```
$ less data/000/001
Received: from unknown (HELO groucho.cs.psu.edu) ([222.135.252.194]) by groucho.cs.psu.edu
        with SMTP id <2533>; Sat, 3 Apr 1993 14:35:34 -0400
Received: from auditor

...

Content-Transfer-Encoding: 7Bit

 LUXURY  WATCHES - BUY YOUR OWN ROLEX  FOR ONLY  $219!

Rolex :: Cartier ::  Bvlgari  ::  Frank Muller  ::  Patek  Philippe :: Vacheron Constantin
A.  Lange  & Sohne  ::  Audemars  Piguet  :: Jaeger-Lecoultre  ::  IWC  :: Officine  Panerai
Breitling ::  Omega ::  Tag  Heuer

...
```

En este caso, como ya indicaba el fichero `index`, se trata de un fichero de *spam*.

3. Representación *bag-of-words*

Como habéis visto en teoría, la representación *bag-of-words* convierte una colección de documentos, en nuestro caso correos electrónicos, en una matriz entera de dimensionalidad: número de documentos (correos electrónicos, filas) por tamaño de vocabulario (*tokens*, columnas).

Como habréis podido comprobar, la representación *bag-of-words* requiere una gran cantidad de memoria si se representa por una matriz completa (no dispersa). Por ello, en

está práctica, haremos uso de matrices dispersas para su representación. Utilizaremos las capacidades de Octave para manipular matrices dispersas e implementar el clasificador multinomial.

Para conocer el formato de matriz dispersa que utiliza Octave lo haremos mediante un ejemplo práctico. Ejecutamos desde la línea de comandos de una terminal:

```
$ octave -q
octave:1>
```

Seguidamente, introduciremos una matriz de ejemplo en formato no disperso, convertiremos esa matriz a formato disperso mediante la función `sparse` (en oposición a la función `full`) y guardaremos a fichero de texto la matriz dispersa resultante. Después ya podemos cerrar la sesión Octave.

```
octave:1> a = [0 5 0; 6 0 0; 7 0 8]
a =

   0   5   0
   6   0   0
   7   0   8

octave:2> data=sparse(a)
data =

Compressed Column Sparse (rows = 3, cols = 3, nnz = 4 [44%])

  (2, 1) ->  6
  (3, 1) ->  7
  (1, 2) ->  5
  (3, 3) ->  8

octave:3> save("-text","sparse.txt","data")
octave:4> quit
```

Podemos analizar el formato de matriz dispersa sobre el fichero de ejemplo "`sparse.txt`".

```
$ cat sparse.txt
# Created by Octave 3.2.3, Tue Feb 26 10:59:56 2013 CET <USER@COMPUTER>
# name: data
# type: sparse matrix
# nnz: 4
# rows: 3
# columns: 3
2 1 6
3 1 7
```

```
1 2 5
3 3 8
```

El fichero presenta la cabecera usual, pero indicando que el tipo de los datos es matriz dispersa (`# type: sparse matrix`), y a partir de la cuarta línea se muestra información específica de matriz dispersa. Empezando por la cabecera, primero, el número de elementos que no son cero `nnz`. Seguidamente el número de filas y columnas, `rows` y `columns`, respectivamente. Finalmente, la representación dispersa de los datos que almacena la matriz en formato índice de fila, índice de columna y valor para esa posición fila-columna. Una vez más es muy importante observar que la matriz dispersa se genera a partir de la matriz densa recorrida por columnas. Es decir, primero recorremos por orden todas las filas de la primera columna, después todas las filas de la segunda columna, etc.

La obtención de la representación como matriz dispersa del conjunto de correos resulta laboriosa, por lo que en esta práctica optamos por proporcionaros estos **correos directamente en formato de matriz dispersa de Octave disponibles en un fichero en PoliformaT (`trec06p.dat.gz`)**. Los correos electrónicos están dispuestos por filas. La última columna de la matriz dispersa indica si el correo electrónico correspondiente a esa fila es *ham* (0) o *spam* (1).

3.1. Ejercicios no entregables

1. Calcula el tamaño de vocabulario, es decir, el número de *tokens* diferentes del corpus *trec06p*.

2. Calcula el espacio en memoria que aproximadamente necesitaría una representación *bag-of-words* en forma de matriz no dispersa basada en un tipo `integer` de 4 bytes.

3. Ídem que el anterior pero en forma de matriz dispersa teniendo en cuenta que es necesario almacenar los índices fila y columna con tipo `integer` de 4 bytes junto con el valor correspondiente.

4. Implementación de un clasificador multinomial

En esta sección implementaremos el entrenamiento de los parámetros de un clasificador multinomial y aplicaremos dicho clasificador al corpus *trec06p*. Al igual que en la sección anterior iremos paso a paso incrementando la complejidad de nuestro desarrollo.

El entrenamiento de los parámetros de un clasificador multinomial está descrito en el tema 5 de teoría en su página 26. Debes tener este tema a mano para la implementación del clasificador multinomial.

4.1. Lectura de datos

El primer paso es crear un *script* Octave donde incluiremos el código a desarrollar. Este *script* recibirá como parámetro el fichero de datos mencionado en la sección anterior y lo cargará en memoria. Por ejemplo:

```
 1 #!/usr/bin/octave -qf
 2
 3 if (nargin!=1)
 4  printf("Usage: multinomial.m <data_filename>");
 5  exit(1);
 6 end
 7
 8 arglist=argv();
 9 datafile=arglist{1};
10 disp("Loading data...");
11 load(datafile);
12 disp("Data load complete.");
```

A partir de ese momento habría cargado en memoria una matriz dispersa `data` que representa las cuentas de los *tokens* del corpus *trec06p*. Como observarás, teniendo la representación en forma de matriz dispersa, la carga en Octave de dicha matriz es trivial.

4.2. Diseño de experimento

Para evaluar el clasificador multinomial que desarrollaremos, necesitamos reservar una parte de nuestro conjunto de datos como conjunto de test. Para ello realizaremos una partición aleatoria de dicho conjunto dedicando 90 % para entrenamiento y 10 % para test. Antes de realizar esta partición es necesario barajar los datos para garantizar que la distribución de clases en entrenamiento y test es similar. Todo ello lo podemos hacer con las siguientes instrucciones Octave:

```
13 [nrows,ncols]=size(data);
14 rand("seed",23);
15 perm=randperm(nrows);
16 pdata=data(perm,:);
17
18 trper=0.9;
19 ntr=floor(nrows*trper);
20 nte=nrows-ntr;
21 tr=pdata(1:ntr,:);
22 te=pdata(ntr+1:nrows,:);
```

Las líneas 14-16 realizan la aleatorización de las filas de la matriz dispersa. La función `size` devuelve el número de filas (número de correos electrónicos) y columnas (número de *tokens* diferentes). La función `randperm` devuelve un vector aleatorio de índices de tamaño `nrows` que se utiliza para aleatorizar las filas de `data` como `pdata`.

Las líneas 18-22 realizan la partición tomando el 90 % de los datos para entrenamiento (`tr`) y el 10 % para test (`te`).

4.3. Entrenamiento del clasificador por máxima verosimilitud

Instanciando las ecuaciones de entrenamiento de un clasificador multinomial (tema 5) a nuestro caso concreto, tenemos N muestras de entrenamiento aleatoriamente extraídas de 2 distribuciones multinomiales independientes, *ham* (h) y *spam* (s)

$$\{(\boldsymbol{x}_n, c_n)\}_{n=1}^{N} \quad \text{i.i.d.} \quad p(\boldsymbol{x}, c) = p(c)\, p(\boldsymbol{x}|c), \quad p(\boldsymbol{x}|c) \sim Mult_D(x_+, \boldsymbol{p}_c) \tag{1}$$

donde $c \in \{h, s\}$ y D es igual al tamaño del vocabulario. El vector de parámetros desconocidos incluye las probabilidades *a priori* (*priors*) y los prototipos multinomiales de cada clase

$$\boldsymbol{\Theta} = (p(h), p(s); \boldsymbol{p}_h, \boldsymbol{p}_s) \tag{2}$$

De acuerdo al criterio de *máxima verosimilitud*, estimamos $\boldsymbol{\Theta}$ como:

$$\hat{p}(h) = \frac{N_h}{N} \qquad\qquad \hat{p}(s) = \frac{N_s}{N} \tag{3}$$

$$\hat{\boldsymbol{p}}_h = \frac{1}{\sum\limits_{n\,:\,c_n=h} \sum\limits_{d} x_{nd}} \sum_{n\,:\,c_n=h} \boldsymbol{x}_n \qquad\qquad \hat{\boldsymbol{p}}_s = \frac{1}{\sum\limits_{n\,:\,c_n=s} \sum\limits_{d} x_{nd}} \sum_{n\,:\,c_n=s} \boldsymbol{x}_n \tag{4}$$

donde $\boldsymbol{x}_n$ es el vector de ocurrencias de palabras en el documento n-ésimo y x_{nd} el número de ocurrencias de la palabra d en el documento n-ésimo.

4.3.1. Ejercicios

1. Teniendo en cuenta que las muestras de entrenamiento están en la variable `tr`, implementa la estimación de las probabilidades *a priori* $p(h)$ y $p(s)$. Para una implementación eficiente utiliza la función `find` que extrae los índices de las filas correspondientes a las muestras de cada clase.

2. Implementa la estimación de los prototipos multinomiales de cada clase. Para una implementación eficiente, evita el uso de instrucciones iterativas y reemplázalas por operaciones matriciales.

4.4. Clasificación multinomial

Ahora que disponemos de los parámetros multinomiales entrenados, no queda más que insertar dichos parámetros en un clasificador multinomial para empezar a clasificar muestras del conjunto de test como *ham* o *spam*.

Una vez más tomamos como referencia los apuntes del tema 5, para definir nuestro clasificador multinomial:

$$c^*(\boldsymbol{x}) = \operatorname*{argmax}_{c \in \{h,s\}} \; g_c(\boldsymbol{x}) \tag{5}$$

con

$$g_h(\boldsymbol{x}) = \boldsymbol{w}_h\, \boldsymbol{x} + w_{h0} \qquad g_s(\boldsymbol{x}) = \boldsymbol{w}_s\, \boldsymbol{x} + w_{s0} \tag{6}$$

donde

$$\boldsymbol{w}_h = \log \boldsymbol{p}_h \qquad \boldsymbol{w}_s = \log \boldsymbol{p}_s \tag{7}$$

$$w_{h0} = \log p(h) \qquad w_{s0} = \log p(s) \tag{8}$$

4.4.1. Ejercicios

En estos dos ejercicios proponemos implementar un clasificador multinomial de dos clases (*ham* y *spam*). El primer ejercicio considera una versión sencilla e ineficiente del clasificador multinomial donde las muestras del conjunto de test son procesadas una a una. El segundo ejercicio considera una implementación matricial más sofisticada y eficiente en que todas las muestras de test son procesadas conjuntamente, tanto en el proceso de clasificación como en la estimación del error empírico. El único objetivo del primer ejercicio es realizar una transición suave hacia la implementación del segundo ejercicio, que es la implementación que realmente utilizaremos en esta práctica. Si consideras que el primer ejercicio es trivial, puedes pasar directamente al segundo ejercicio.

1. Considera una implementación del clasificador multinomial en la que las muestras del conjunto de test (`te`) son clasificadas una a una (bucle `for`). Calcula el error cometido por el clasificador multinomial comparando las etiquetas de clase estimadas por el mismo y las etiquetas de clase reales (última columna de `te`).

2. Refina tu implementación anterior de forma que la clasificación y estimación del error empírico se realicen mediante operaciones matriciales.

4.5. Suavizado de parámetros multinomiales

Habrás observado que la ejecución del anterior clasificador proporciona unos resultados de clasificación pésimos (error superior al 50 %). Esto es debido a que nuestros prototipos multinomiales contienen probabilidades nulas para algunos *tokens*. Hay *tokens* que nunca aparecen en los correos electrónicos de la clase *ham* y algunos que no lo hacen para la clase *spam*.

Para aliviar este problema aplicaremos el *suavizado de Laplace* (ver tema 5 de teoría) sobre nuestros prototipos multinomiales.

$$\tilde{\boldsymbol{p}}_{cd} = \frac{\hat{\boldsymbol{p}}_{cd} + \epsilon}{\sum\limits_{d} (\hat{\boldsymbol{p}}_{cd} + \epsilon)} \tag{9}$$

donde $\epsilon > 0$ y $c \in \{h, s\}$. En la práctica, basta con sumar el valor de ϵ a los valores calculados de $\hat{\boldsymbol{p}}_h$ y $\hat{\boldsymbol{p}}_s$ y normalizar adecuadamente.

4.5.1. Ejercicios

1. Implementa el suavizado de Laplace teniendo en cuenta que ϵ es un *hiperparámetro* que proporcionarás en cada ejecución al clasificador multinomial.

2. Calcula el error de clasificación considerando que $\epsilon = 0{,}1$. El error obtenido debería estar entorno al 19,1 %.

3. Calcula el error de clasificación considerando que $\epsilon = 0{,}0001$. El error obtenido debería ser entorno al 7,6 %.

4. Como observarás, el error de clasificación varía significativamente en función del valor del hiperparámetro de suavizado ϵ. Estudia el comportamiento del error de clasificación en función del hiperparámetro $\epsilon = 10^{-1}, 10^{-2}, \ldots, 10^{-20}$.

5. Representa gráficamente los resultados obtenidos teniendo en cuenta que seguramente sea necesario aplicar escalado logarítmico tanto al eje X como al eje Y.

6. Las tasas de error calculadas hasta el momento están obtenidas sobre una única partición aleatoria de los datos y por tanto no son suficientemente robustas ya que no disponemos de intervalos de confianza para los errores calculados. Para solventar este problema, por cada valor del hiperparámetro ϵ realizaremos 30 repeticiones del proceso de partición aleatoria y calcularemos la media y desviación típica.

7. Representa gráficamente las tasas de error del anterior ejercicio utilizando intervalos de confianza del error (*errorbars*) al 95 % (error medio más menos 2 desviaciones típicas del error calculado sobre 30 repeticiones del mismo ϵ). El resultado deberá ser semejante al que propone la Figura 1.

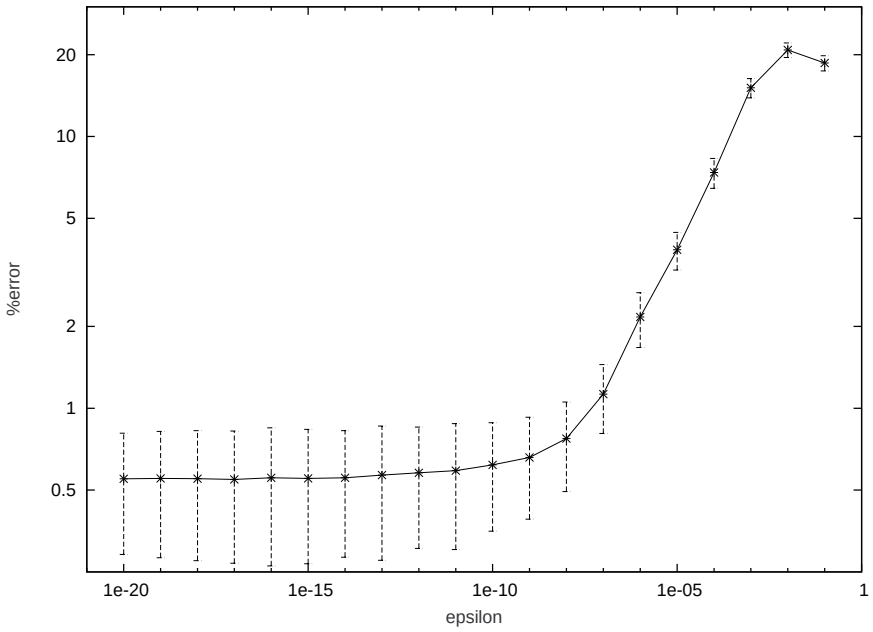

Figura 1: Error del clasificador multinomial en función del hiperparámetro ϵ.

5. Ejercicios adicionales (opcional no evaluable)

1. Cuando estudiamos la distribución multinomial $Mult_D(x_+, \boldsymbol{p})$ en clase consideramos que la longitud de la secuencia x_+ es un valor constante para todas las muestras. Sin embargo, en esta práctica hemos obviado este detalle entrenando y clasificando correos electrónicos con diferente longitud (número de tokens). Implementa y evalua una modificación de tu código de forma que todas las muestras utilizadas en el entrenamiento y clasificación tengan la misma longitud.

2. En esta práctica hemos considerado desde un principio que el 10 % del conjunto de datos estaría destinado al test de parámetros y el 90 % restante al conjunto de entrenamiento. Sin embargo, sería interesante estudiar el comportamiento de nuestro clasificador multinomial en función del porcentaje de datos de entrenamiento disponibles. Evalúa el error de clasificación cuando el tamaño del conjunto de entrenamiento varía de acuerdo a los siguiente porcentajes 1 %, 2 %, 5 %, 10 %, 20 % y 50 %. Representa los resultados obtenidos como múltiples curvas sobre la gráfica obtenida anteriormente.

3. El método de partición usado en el ejercicio previo puede tener un problema de sesgo (*bias*) que se suaviza con el número de repeticiones aleatorias. Otra técnica experimental es la validación cruzada (*cross-validation*), donde el conjunto total de datos se divide en n bloques de tamaño semejante y se toma un bloque para test y los $n-1$ restantes para entrenamiento. El proceso se repite para todos los bloques, dando como resultado un experimento donde todos los datos disponibles han pasado por entrenamiento y test, garantizando un menor sesgo. Implementa la experimentación por validación cruzada empleando 10 particiones de tamaño similar y dibuja la gráfica de resultados correspondiente.

4. Considera el caso en que hubiéramos optado por una representación binaria de *trec06p* en la que únicamente indicáramos si un *token* está presente en un correo electrónico o no. Bajo este supuesto podríamos utilizar un clasificador Bernoulli (ver tema 5 de teoría).

   a) Implementa el entrenamiento de los parámetros de un clasificador Bernoulli por máxima verosimilitud.

   b) Implementa el clasificador Bernoulli.

   c) Implementa el suavizado por truncamiento simple.

   d) Estudia el comportamiento del error del clasificador Bernoulli bajo las mismas particiones entrenamiento y test. Explora los valores de ϵ para el suavizado teniendo en cuenta que serán diferentes.

   e) Representa gráficamente los resultados obtenidos al igual que hiciste para el clasificador multinomial.

   f) Compara el comportamiento del clasificador Bernoulli frente al multinomial.

6. Evaluación de la práctica

La práctica supone un 15 % del total de la nota de la asignatura (1.5 puntos). Para su evaluación se tendrán en cuenta dos actividades:

1. Una memoria del trabajo básico pedido en esta práctica, con los siguientes contenidos:

   - Código convenientemente comentado de `multinomial.m` (0.5 puntos)
   - Gráfica de resultados comparativos (0.25 puntos)
   - Comentario de los resultados (0.25 puntos)

   El día de entrega de la memoria es el **27 de mayo**.

2. Una competición que se desarrollará sobre otro corpus de datos donde se buscará el mejor clasificador, ya sea basado en multinomial o en Bernoulli, con distintos suavizados. Esta actividad se realizará durante la última sesión de prácticas (0.5 puntos máximo, dependiendo de la tasa de error de clasificación obtenida).