Clasificadores de documentos: Naive Bayes y Modelo Bernoulli

Daniel A. Mesejo León, Victor M. Mendiola Lau, Yassiel E. Oliva Carmona

Facultad de Matemática y Computación, Universidad de la Habana,
Ciudad de La Habana, Cuba.
{d.mesejo,v.mendiola,y.oliva}@lab.matcom.uh.cu

Resumen En este trabajo se realiza un estudio de los clasificadores de documentos *Naive Bayes* y el *Modelo Bernoulli*. Se analizan sus características fundamentales, así como sus semejanzas y diferencias. Se implementaron los selectores de características: *Mutual Information*, χ^2 y *Frequency-based*. Con el objetivo de realizar un análisis más exhaustivo, se realizaron comparaciones con clasificadores conocidos como: *KNN* y *Roccio*.

Keywords: Naive Bayes, Modelo Bernoulli, clasificación de documentos

1. Introducción

El objetivo principal de la clasificación de documentos es determinar, dado un conjunto de clases o categorías y un conjunto de documentos, qué categoría(s) se ajusta mejor al contenido de un documento especifíco basándose en los términos del mismo.

2. Clasificación de documentos utilizando Naive Bayes

Naive Bayes (NB) [2] es un algoritmo de aprendizaje supervisado. En este, la probabilidad de que un documento d pertenezca a una clase c se denota como $P(c|d)$ y se puede calcular de manera aproximada como se muestra en la ecuación (1).

$$P(c|d)\alpha P(c)\prod_{i=1}^{n} P(t_k|c) \tag{1}$$

Donde $P(t_k|c)$ es la probabilidad condicional de que el término t_k aparezca en un documento de clase c, $P(t_k|c)$ puede ser tomada como una medida de cuánta evidencia indica que c es la clase correcta para d. El término $P(c)$ representa la probabilidad a priori de que un documento pertenezca a la clase c. Si los términos de un documento no muestran una clara evidencia de preferencia entre dos o más categorías, se elegirá de la mayor $P(c)$. En *NB*, la clase que más se ajusta a un documento, será la clase máxima a posteriori (max), se denotará como *c-max* y se determinará como aparece en la ecuación (2).

$$c\text{-}max = argmax_{c\in C}\hat{P}(c|d) = argmax_{c\in C}\hat{P}(c)\prod_{1\leq k\leq n_d}\hat{P}(t_k|c) \tag{2}$$

Aquí se utilizó $\hat{P}$ en lugar de P ya que los valores reales de los parámetros $P(c)$ y $P(t_k|c)$ no son conocidos, peros serán estimados a partir del conjunto de prueba. Como podemos notar en la ecuación (2) se realiza una gran cantidad de multiplicaciones de probabilidades condicionales, lo que puede traer problemas de redondeo en la aritmética de punto flotante. Por esta razón, utilizando las propiedades de los logaritmos, obtendremos una ecuación equivalente, ya que se conserva la monotonía de la función. En la ecuación (3) se muestra la maximización más utilizada en las implementaciones de *NB*.

$$c\text{-}max = argmax_{c\in C}\log\hat{P}(c) + \sum_{1\leq k\leq n_d}\log\hat{P}(t_k|c) \tag{3}$$

En la ecuación (3), $\hat{P}(t_k|c)$ es un peso para ver cuán bueno es el indicador t_k para c. La suma del logaritmo de la probabilidad a priori con el peso de los términos servirá de medida para decir cuánta evidencia existe para decir que el documento d pertenece a la clase c.

Ahora estamos en condiciones de conocer cómo se estimarán los valores de $\hat{P}(c)$ y $\hat{P}(t_k|c)$. En el caso de la probabilidad a priori, realizaremos una aproximación mediante la frecuencia relativa, esto se puede apreciar en la ecuación (4).

$$\hat{P}(c) = \frac{N_c}{N} \tag{4}$$

Donde N_c representa el número de documentos en la clase c y N el total de documentos de la colección. En el caso de la probabilidad condicional $\hat{P}(t_k|c)$, la estimación se realizará con la frecuencia relativa del término t en los documentos de la clase c, ver (5).

$$\hat{P}(t_k|c) = \frac{T_{ct}}{\sum_{p \in V} T_{cp}} \tag{5}$$

Donde V es el vocabulario y T_{ct} es el número de ocurrencias del término t en los documentos de la clase c, incluyendo las múltiples apariciones de un término dentro de los documentos. Como consecuencia de esto, si un término aparece en un documento en las posiciones k_1 y k_2, entonces se cumplirá que $\hat{P}(t_{k_1}|c) = \hat{P}(t_{k_2}|c)$. El problema de estas aproximaciones se puede ver analizando la ecuación (5), porque en caso que un término no aparezca en ninguna clase, $\hat{P}(t_k|c)$ sería 0 y el logaritmo se indefine. Para lidiar con los ceros realizaremos la modificación *add-one*, que simplemente añade uno a cada miembro. La ecuación (6) muestra dicha transformación.

$$\hat{P}(t_k|c) = \frac{T_c t + 1}{\sum_{p \in V} T_{cp} + B} \tag{6}$$

Donde $B = |V|$. Esta aproximación puede ser vista como que cada término ocurre al menos una vez en cada categoría. Ahora veamos un pseudocódigo del algoritmo en la fase de entrenamiento (**Algoritmo 1**) y la fase de clasificación (**Algoritmo 2**).

Algorithm 1: Entrenamiento

input : Un conjunto de clases C y un conjunto de documentos D
output: el vocabulario de los documentos y las probabilidades de los términos de estar en una clase determinada

```
1  V ← ExtractVocabulary(D) ;
2  N ← CountDocs(D) ;
3  for c ∈C do
4  |   N_c ← CountDocsInClass(D, c) ;
5  |   prior [c] ← N_c/N ;
6  |   text_c ← ConcatenateTextOfAllDocsInClass(D, c);
7  |   for t ∈V do
8  |   |   T_c t ← CountTokensOfTerm(text_c, t) ;
9  |   end
10 |   for t ∈V do
11 |   |   condprob [t][c] ← (T_c t+1)/(Σ_{p∈V} T_cp) ;
12 |   end
13 |   return V, prior, condprob
14 end
```

Algorithm 2: Clasificación

input : C, V, prior, condprob, d
output: la clase que más se ajusta al documento

```
1  V ← ExtractTokensFromDoc(V, d) ;
2  for c ∈C do
3  |   score [c] ← log prior[c] ;
4  |   for t ∈W do
5  |   |   score [c] += logcondprob[t][c] ;
6  |   end
7  |   return argmax_{c∈C} score[c]
8  end
```

2.1. Complejidad temporal

La complejidad de computar los parámetros es $\Theta(|C||V|)$, ya que el conjunto de parámetros está compuesto por $|C|$ probabilidades a priori y $|C||V|$ probabilidades condicionales [2]. El preprocesamiento necesario para calcular los parámetros (extracción de vocabulario, conteo de términos, etc.) se puede lograr en una pasada por el conjunto de datos [2]. Por tanto, la complejidad temporal es de $\Theta(|D|L)$, donde D es el conjunto de documentos y L es el promedio

de las longitudes de los documentos. La complejidad temporal del proceso de clasificación sería $\Theta(|C|L_a)$, donde L_a es la cantidad de términos.

3. Modelo Bernoulli

El *Modelo Bernoulli* (también conocido como modelo binomial multivariado) [2] es una de las alternativas al modelo multinomial. La principal diferencia respecto al modelo multinomial es que el modelo Bernoulli no toma en cuenta la cantidad de ocurrencias de un término dentro de un documento, este guarda un indicador para cada término, que toma valor 1 en caso de que el término ocurra dentro del documento y valor 0 en otro caso. Como consecuencia de la diferencia que existe en la generación de los modelos, la manera en la que se estimarán los parámetros cambia igualmente. El *Modelo Bernoulli* estimará $P(t_k|c)$ como con la fracción de la línea 8 del **Algoritmo 3**.

El hecho de no considerar la cantidad de apariciones de un término dentro de un documento en la clasificación, trae como consecuencia que la única aparición de un término en un documento, condicione la categoría que le será asignada. Este modelo comete varios errores en colecciones con documentos largos por lo que no es aconsejable usarlo en estas situaciones.

Algorithm 3: Entrenamiento

input : Un conjunto de clases C y un conjunto de documentos D
output: el vocavulario de los documentos y las probabilidades de los términos de estar en una clase determinada

```
1  V ← ExtractVocabulary (D) ;
2  N ← CountDocs(D) ;
3  for c ∈C do
4  |   N_c ← CountDocsInClass(D, c) ;
5  |   prior [c] ← N_c/N ;
6  |   for t ∈V do
7  |   |   N_c t ← CountDocsInClassContainingTerm(D, c, t) ;
8  |   |   condprob [t][c] ← (N_c t+1)/(N_c+2) ;
9  |   end
10 |   return V, prior, condprob
11 end
```

3.1. Complejidad temporal

El *Modelo Bernoulli* presenta un coste temporal similar al analizado en *Naive Bayes*.

Algorithm 4: Clasificación

input : C, V, prior, condprob, d
output: la clase que más se ajusta al documento

```
1  V_d ← ExtractTermsFromDoc(V, d) ;
2  for c ∈C do
3  |  score [c] ← log prior[c] ;
4  |  for t ∈V do
5  |  |  if t ∈ V_d then
6  |  |  |  score [c] += logcondprob[t][c];
7  |  |  else
8  |  |  |  score [c] += log1 − condprob[t][c] ;
9  |  |  end
10 |  end
11 |  return argmax_{c∈C} score[c]
12 end
```

4. Detalles de la implentación

La implementación de estos algoritmos fue desarrollada completamente en Java utilizando como IDE IntelliJ IDEA 11.1.2 de Jetbrains. Para la manipulación de la colección de documentos de prueba, se utilizó Lucene [1].

5. Resultados

Para la realización de las pruebas se utilizó la colección de documentos reuters21578, de esta se seleccionaron un conjunto de 178 documentos por dos razones fundamentales: existían documentos sin contenido (estos afectaban la calidad del entrenamiento) y existían dos grandes grupos que contribuían a un entrenamiento sobreespecializado.

A fin de tener una idea clara de la efectividad de los algoritmos NB y Bernoulli, se realizaron comparaciones con los clasificadores KNN y Roccio. En la tabla 1 se muestran los resultados obtenidos, seleccionando de manera aleatoria distintos conjuntos de prueba. Los números se obtuvieron utilizando la medida de evaluación *cross-validation*, donde mientras más cercano a 1 se encuentre el resultado, mejor fue la clasificación.

Entrenamiento	*Prueba*	*Bernoulli*	*NaiveBayes*	$KNN(3)$	$KNN(5)$	*Roccio*
150	28	0.2281	0.8104	0.4676	0.4247	0.2102
130	48	0.2473	0.7579	0.4246	0.4038	0.1574
100	78	0.2572	0.7458	0.4687	0.4162	0.1664
90	88	0.2643	0.736	0.4144	0.3711	0.1566

Tabla 1. Tabla comparativa entre los algortimos

5.1. Selección de características

El proceso de selección de características hace referencia a seleccionar un subconjunto de términos del conjunto de términos de los documentos entrenantes y realizar el entrenamiento con este subconjunto. La selección de características tiene dos objetivos principales. Primero, hace que la aplicación y entrenamiento de un clasificador sea más eficiente, pues reduce la longitud del vocabulario. Segundo, a menudo incrementa la precisión de la clasificación, pues elimina características ruidosas.

Una característica ruidosa, es aquella que incrementa el error cuando se añade a la representación de un documento. Supongamos un término raro, por ejemplo *cucaracha*, el cual no contiene ninguna información sobre la clase *Cuba*, pero sucede que todas las instancias de *cucaracha* ocurren en *Cuba*. Un clasificador puede añadir por error un documento con el término *cucaracha* a la clase *Cuba*. Para seleccionar varias de estas características se han diseñado varios selectores, entre ellos *Mutual Information*, χ^2 y *Frequency-based*. Para una profundización en este tema se remite al lector a [2].

Entrenamiento	*Prueba*	$B(CS)$	$NB(CS)$	$B(MI)$	$NB(MI)$	$B(F)$	$NB(F)$
150	28	0.267	0.696	0.274	0.617	0.257	0.642
130	48	0.231	0.654	0.239	0.641	0.222	0.647
100	78	0.269	0.634	0.219	0.614	0.192	0.575
90	88	0.192	0.622	0.219	0.634	0.190	0.645

Tabla 2. Resultados con diferentes selectores de características. CS representa a la selección χ^2, MI a la *Mutual Information* y F a un selector basado en la frecuencia de los términos.

6. Conclusiones

Después de realizar pruebas con ambos algoritmos, se puede afirmar que el *Multinomial Naive Bayes* resulto ser más efectivo que el *Modelo Bernoulli*. Esto se debe a que este tiene en cuenta la cantidad de ocurrencias de los términos

dentro de los documentos, con lo cual logra un mejor porcentaje de efectividad al probar con colecciones de documentos grandes.

El *Modelo Bernoulli* tiene entre sus deficiencias que puede clasificar erróneamente un documento, debido que solo tienen en cuenta la ocurrencia o no ocurrencia de los términos en cada uno de los documentos. Otra observación es que existen condiciones bajo las cuales el Naive Bayes Multinomial es superior a KNN, uno de los algoritmos básicos de la clasificación de documentos. La fácil implementación y gran efectividad de el *Naive Bayes Multinomial* lo hacen un algoritmo muy atractivo a la hora de clasificar documentos.

Referencias

1. Erick Hatcher, Otis Gospodnetic, and Michael McCandless. *Lucene In Action*. Manning Publications, 2009.
2. Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *An Introduction to Information Retrieval*. Cambridge University Press, 2007.