Agrupamiento y clasificación en la recuperación de información en la web

Integrantes:
Marcos Manuel Tirador del Riego
Laura Victoria Riera Pérez

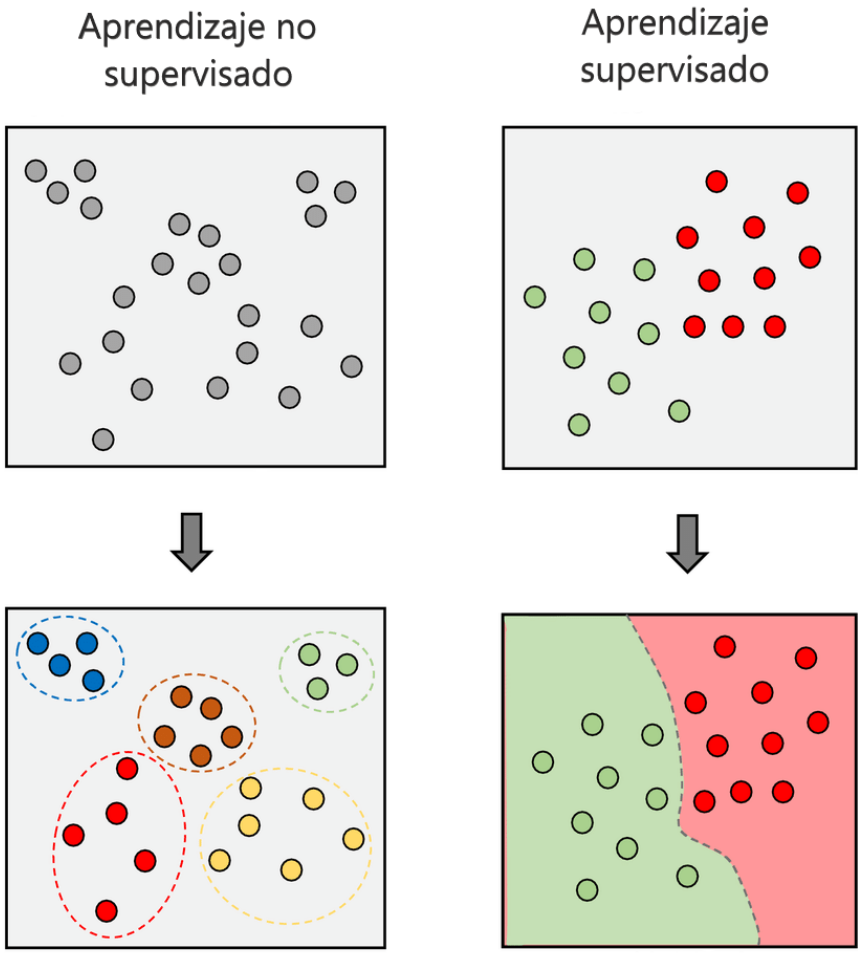
Tercer aÃśo. Ciencias de la computación. Universidad de La Habana. Cuba.

Noviembre, 2022

Índice general I

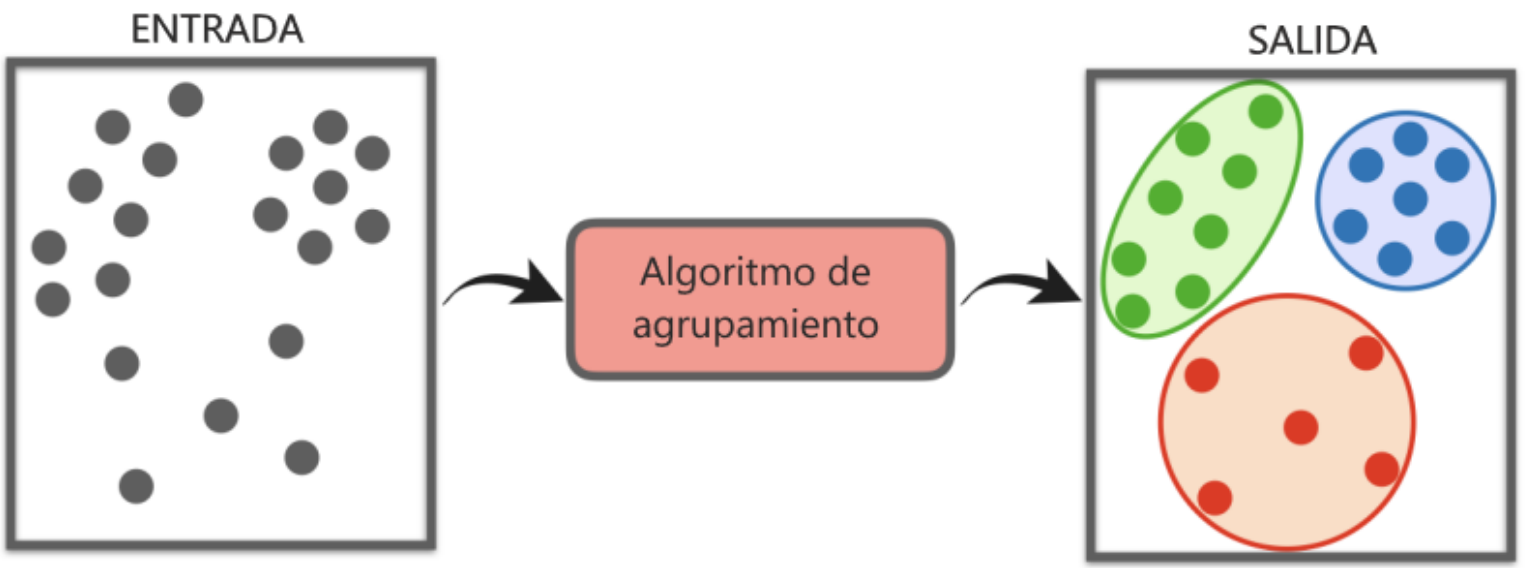
Índice general II

Aprendizaje no supervisado vs. aprendizaje supervisado



Agrupamiento I

Los algoritmos de agrupamiento conglomeran un conjunto de documentos en subconjuntos o clústeres. Son utilizados para generar una estructura de categorÃŋas que se ajuste a un conjunto de observaciones.



Características generales

- Es la forma mÃąs comÃžn de aprendizaje no supervisado.
- Los grupos formados deben tener un alto grado de asociaciÃșn entre los documentos de un mismo grupo y un bajo grado entre miembros de diferentes grupos.
- La entrada clave para un algoritmo de agrupamiento es la medida de distancia. Diferentes medidas de distancia dan lugar a diferentes agrupamientos.

Hipótesis de agrupamiento

"Los documentos en el mismo grupo se comportan de manera similar con respecto a la relevancia para las necesidades de informaciÃșn."

La hipÃștesis establece que si hay un documento de un grupo que es relevante a una solicitud de bÃžsqueda, entonces es probable que otros documentos del mismo clÃžster tambiÃ‘n sean relevantes.

Clasificación de los algoritmos de agrupamiento

Según la pertenencia a los grupos:

- *agrupamiento exclusivo o fuerte* (hard clustering): cada documento es miembro de exactamente un grupo.
- *agrupamiento difuso o suave* (soft clustering): un documento tiene membresÃŋa fraccionaria en varios grupos.

Según el tipo de estructura impuesta sobre los datos:

- *agrupamiento particionado o plano* (flat clustering)
- *agrupamiento jerárquico* (hierarchical clustering).

Medidas de similitud

Sean d_i el i-Ãľsimo documento del corpus y w_{ik} el peso del término k de un total N $(N > 0)$ en este documento.

- **Coeficiente de Dice:**

$$S_{d_i,d_j} = \frac{2\sum_{k=1}^{N}(w_{ik}w_{jk})}{\sum_{k=1}^{N} w_{ik}^2 + \sum_{k=1}^{N} w_{jk}^2}$$

- **Coeficiente de Jaccard:**

$$S_{d_i,d_j} = \frac{\sum_{k=1}^{N}(w_{ik}w_{jk})}{\sum_{k=1}^{N} w_{ik}^2 + \sum_{k=1}^{max} w_{jk}^2 - \sum_{k=1}^{N}(w_{ik}w_{jk})}$$

- **Coeficiente del coseno:**

$$S_{d_i,d_j} = \frac{\sum_{k=1}^{N}(w_{ik}w_{jk})}{\sqrt{\sum_{k=1}^{N} w_{ik}^2 \sum_{k=1}^{N} w_{jk}^2}}$$

Medidas de evaluación I

- **Pureza**
- **InformaciÃșn mutua normalizada**
- **Índice de Rand (Rand Index)**
- **Medida F**

Agrupamiento particionado

El agrupamiento particionado crea un conjunto de clÃžsteres sin ninguna estructura explÃŋcita que los relacione entre sÃŋ.

K-means I

Es el algoritmo de agrupamiento particionado mÃąs importante. Su objetivo es minimizar la distancia euclidiana al cuadrado promedio entre los documentos y el centro de sus clústeres.
El centro de un clúster se define como la media o centroide μ de los documentos en un grupo ω:

$$\vec{\mu}(\omega) \leftarrow \frac{1}{|\omega_k|} \sum_{\vec{x} \in \omega_k} \vec{x}$$

K-means II

Una medida de quÃľ tan bien los centroides representan a los miembros de su clúster es la suma residual de cuadrados o RSS, que es la distancia al cuadrado de cada vector desde su centroide sumado sobre todos los vectores:

$$RSS_k = \sum_{\vec{x} \in \omega_k} |\vec{x} - \vec{\mu}(\omega_k)|^2$$

$$RSS = \sum_{k=1}^{K} RSS_k$$

RSS es entonces la funciÃșn objetivo en K-means y nuestro objetivo es minimizarla.

K-means III

Algoritmo 1 K-Means

Parámetros de entrada: $\{\overrightarrow{x_1}, ..., \overrightarrow{x_N}\}, K$

1: $(\overrightarrow{s_1}, ..., \overrightarrow{s_K}) \leftarrow$ **SelectRandomSeeds**$(\{\overrightarrow{x_1}, ..., \overrightarrow{x_N}\}, K)$
2: **for** $k \leftarrow 1$ **to** K **do**
3: $\quad \overrightarrow{\mu_k} \leftarrow \overrightarrow{s_k}$
4: **while** no se cumpla la condición de parada **do**
5: $\quad$ **for** $k \leftarrow 1$ **to** K **do**
6: $\quad\quad \omega_k \leftarrow \{\}$
7: $\quad$ **for** $n \leftarrow 1$ **to** N **do**
8: $\quad\quad j \leftarrow \arg\min_{j'} |\overrightarrow{\mu_{j'}} - \overrightarrow{x_n}|$
9: $\quad\quad \mu_j \leftarrow \omega_j \cup \{\overrightarrow{x_n}\}$ (reasignando los vectores)
10: $\quad$ **for** $k \leftarrow 1$ **to** K **do**
11: $\quad\quad \overrightarrow{\mu_k} \leftarrow \frac{1}{|\omega_k|} \sum_{\overrightarrow{x} \in \omega_k} \overrightarrow{x}$ (reeligiendo las semillas)
12: **return** $\{\overrightarrow{\mu_1}, ..., \overrightarrow{\mu_N}\}$

K-means IV

El primer paso de esta implementación de K-means es seleccionar al azar como centros iniciales de los clústeres a K documentos, estas son las semillas. Luego, el algoritmo mueve los centros de los grupos en el espacio para minimizar el RSS. Este proceso se repite de manera iterativa hasta que se cumpla un criterio de parada.

Criterios de parada:

- Cuando se ha completado un nÃžmero fijo de iteraciones I.
- Cuando la asignaciÃșn de documentos a grupos no cambia entre iteraciones.
- Cuando los centroides no cambian entre iteraciones.
- Cuando RSS cae por debajo de un umbral.

Agrupamiento jerárquico

El agrupamiento jerÃąrquico produce una jerarquÃŋa, una estructura que es mÃąs informativa que el conjunto no estructurado de clÃžsteres devuelto por el agrupamiento particionado.

Pueden tener dos enfoques:

- De abajo hacia arriba (bottom-up) llamados de *agrupamiento jerárquico aglomerativo*.
- De arriba hacia abajo (top-down) conocidos como de *agrupamiento jerárquico divisivo*.

Agrupamiento jerárquico aglomerativo

Los algoritmos de abajo hacia arriba tratan cada documento como un clÃžster Ãžnico desde el principio y luego fusionan (o aglomeran) sucesivamente pares de grupos hasta que todos los grupos se han fusionado en uno solo que contiene todos los documentos. Es por esto que se denomina agrupamiento jerÃąrquico aglomerativo o HAC por sus siglas en inglés.

Toman decisiones basadas en patrones locales sin tener inicialmente en cuenta la distribuciÃșn global. Estas decisiones tempranas no se pueden deshacer.

Medidas de similitud para clústeres en HAC I

- **Agrupamiento por enlazamiento único** (single link clustering): La similitud entre dos clústers es la similitud de los dos objetos mÃąs cercanos entre ellos (mayor similitud).
- **Agrupamiento por enlazamiento completo** (complete link clustering): La similitud entre dos clústers es la similitud de los dos objetos mÃąs alejados entre ellos (menor similitud).
- **Agrupamiento aglomerativo por promedio de grupo** (group-average agglomerative clustering): Calcula la similitud promedio de todos los pares de documentos, incluidos los pares del mismo grupo, evitando asÃŋ castigar valores extremos como en los criterios de enlace Ãžnico y enlace completo.
- **Agrupamiento por centroide** (centroid clustering): La similitud de dos clústers está definida como la similitud de sus centroides.

Algoritmo HAC I

Dado un conjunto de N elementos a agrupar, el proceso bÃąsico del agrupamiento jerÃąrquico aglomerativo es:

1. Se comienza con N clústeres, resultado de asignar cada elemento al suyo propio. Se computa la matriz C de similitud de NÃůN.
2. Se halla la similitud entre los pares de clústeres con la medida deseada.
3. Se toma el par mÃąs similar de clÃžsteres y se combinan en un Ãžnico clÃžster.
4. Se calculan las similitudes entre el nuevo clÃžster y cada uno de los clústeres antiguos.
5. Se repiten los pasos 3 y 4 hasta que todos los elementos estÃľn agrupados en un solo grupo de tamaÃśo N.

Algoritmo HAC II

Algoritmo 2 HAC

Parámetros de entrada: $\{d_1, ..., d_N\}$

for $n \leftarrow 1$ **to** N **do**
  for $i \leftarrow 1$ **to** N **do**
    $C[n][i] \leftarrow SIM(d_n, d_i)$
    $I[n] \leftarrow 1$ (lleva los clústeres activos)
  $A \leftarrow []$ (secuencia de mezclas aplicadas en el agrupamiento)
for $k \leftarrow 1$ **to** $N - 1$ **do**
  $< i, m > \leftarrow \arg\max_{<i,m>: i \neq \wedge I[i]=1 \wedge I[m]=1} C[i][m]$
  $A.APPEND(< i, m >)$ (almacena la mezcla)
  for $j \leftarrow 1$ **to** N **do**
    $C[i][j] \leftarrow SIM(i, m, j)$
    $C[j][i] \leftarrow SIM(i, m, j)$
  $I[m] \leftarrow 0$ (desactiva el clúster)
return A

Agrupamiento jerárquico divisivo

Los algoritmos de arriba hacia abajo comienzan con todos los documentos en un grupo. El clÃžster se divide utilizando un algoritmo de agrupamiento particionado. Este procedimiento se aplica recursivamente hasta que cada documento estÃą en su propio clÃžster.

Se beneficia de la informaciÃșn completa sobre la distribuciÃșn global al tomar decisiones de particiÃșn de alto nivel.

Ventajas

- No es necesario identificar las clases antes del procesamiento por lo que no se debe contar con expertos para este fin.
- Es Ãžtil para proporcionar estructura en grandes conjuntos de datos multivariados.
- Se ha descrito como una herramienta de descubrimiento porque tiene el potencial para revelar relaciones previamente no detectadas basadas en datos complejos.
- Debido a su amplia aplicaciÃșn en disímiles campos, cuenta el apoyo de una serie de paquetes de software, a menudo disponibles en la informÃątica acadÃľmica y otros entornos, por lo que se facilita su utilización.

Desventajas

- No se tiene una idea exacta de las clases creadas.
- No recibe retroalimentación.

Ejemplos de aplicación en la web I

El agrupamiento es una tÃľcnica importante para descubrir subregiones o subespacios relativamente densos de una distribuciÃșn de datos multidimensional. Se ha utilizado en la recuperaciÃșn de informaciÃșn para muchos propÃșsitos diferentes, como la expansiÃșn de consultas, la agrupaciÃșn e indexaciÃșn de documentos y la visualizaciÃșn de resultados de bÃžsqueda. Permiten mejorar interfaz y experiencia de usuario y proporcionar una mayor eficacia o eficiencia del sistema de bÃžsqueda.

Ejemplos de aplicación en la web II

- **Agrupamiento de resultados de bÃžsqueda** (Search result clustering): La presentaciÃșn predeterminada de los resultados de bÃžsqueda (documentos devueltos en respuesta a una consulta) en la recuperaciÃșn de informaciÃșn es una lista sencilla. Los usuarios escanean la lista de arriba a abajo hasta que encuentran la informaciÃșn que buscan.
  En su lugar, en la agrupaciÃșn en clÃžsteres de resultados de bÃžsqueda los documentos similares aparecen juntos, siendo mÃąs fÃącil escanear algunos grupos coherentes que muchos documentos individuales. Esto es particularmente Ãștil si un tÃ‘rmino de bÃžsqueda tiene diferentes significados.

Ejemplos de aplicación en la web III

- **DispersiÃșn-recopilaciÃșn** (Scatter-Gather): Su objetivo es también una mejor interfaz de usuario. Este agrupa toda la colecciÃșn para obtener grupos de documentos que el usuario puede seleccionar o reunir manualmente. Los grupos seleccionados se fusionan y el conjunto resultante se vuelve a agrupar. Este proceso se repite hasta que se encuentre un grupo de interÃľs.
  La navegaciÃșn basada en la agrupación de clÃžsteres es una alternativa interesante a la bÃžsqueda por palabras clave, el paradigma de recuperaciÃșn de informaciÃșn estÃąndar. Esto es especialmente cierto en escenarios donde los usuarios prefieren navegar en lugar de buscar porque no estÃąn seguros de quÃľ tÃľrminos utilizar en la bÃžsqueda.

Ejemplos de aplicación en la web IV

- **Recuperación basada en clústeres** (Cluster-based retrieval): La bÃžsqueda en el modelo de espacio vectorial equivale a encontrar los vecinos mÃąs cercanos a la consulta. El Ãŋndice invertido admite la bÃžsqueda rÃąpida del vecino mÃąs cercano para la configuraciÃșn, sin embargo, a veces es posible que no se pueda usar un Ãŋndice invertido de manera eficiente. En tales casos, se podría calcular la similitud de la consulta con cada documento, pero esto es lento. La hipÃștesis de agrupamiento ofrece una alternativa: encontrar los clústeres que estÃąn mÃąs cerca de la consulta y sÃșlo considerar los documentos de estos. Como hay muchos menos clÃžsteres que documentos, se disminuye grandemente el espacio de búsqueda. Además, los elementos que coinciden con una consulta son similares entre sÃŋ, por lo que tienden a estar en los mismos clústeres, de esta forma la calidad no disminuye en gran medida.

Clasificación I

El problema de clasificación en sentido general consiste en determinar dentro de un conjunto de clases a cuál de ellas pertenece un objeto dado. En el marco de este documento es de interés estudiar la clasificación de textos.
La forma más antigua de llevar a cabo la clasificación es manualmente. Por ejemplo, los bibliotecarios clasifican los libros de acuerdo a ciertos criterios, de modo que encontrar una información buscada no resulte una tarea de gigante dificultad. Sin embargo, la clasificación manual tiene sus límites de escalabilidad.

Clasificación II

Como alternativa podría pensarse el uso de *reglas* para determinar automáticamente si un texto pertenece o no a una clase determinada de documentos. Esto está relacionado con las consultas permanentes (o *standing querys* en inglés). Una consulta permanente es aquella que se ejecuta periódicamente en una colección a la cual nuevos documentos se adicionan en el tiempo. Toda consulta permanente se puede ver como un tipo de regla que se aplica a un sistema de clasificación que divide una colección en documentos que satisfacen la consulta y documentos que no.

Una regla captura una cierta combinación de palabras claves que identifican una clase. Reglas codificadas a mano pueden llegar a ser altamente escalable, pero crearlas y mantenerlas requiere un elevado costo en recursos humanos.

Clasificación III

Existe, sin embargo, un enfoque adicional a los dos anteriores mencionados. Este es el uso de *Aprendizaje de Máquinas*. En este enfoque el conjunto de reglas de clasificación, o en general, el criterio usado para clasificar, es aprendido de forma automática a partir de los datos de entrenamiento.
Al uso del Aprendizaje de Máquinas en la clasificación de textos se le conoce como clasificación estadística de texto (o en inglés *statistical text classification*) si el método de aprendizaje usado es estadítico.

Clasificación IV

Se presenta a continuación la definición formal del problema de clasificación de textos, en el contexto del Aprendizaje de Máquinas.

Definición 1: Sea $\mathscr{X}$ el espacio de documentos y $\mathscr{C} := \{c_i \mid c_i \subset \mathscr{X}, i \in \{1, 2, \ldots, n\}\}$ un conjunto fijo de clases (también llamadas categorías o etiquetas). Sea además D un conjunto entrenado de documentos clasificados $(d, c) \in \mathscr{X} \times \mathscr{C}$. El *problema de la clasificación de textos* consiste en encontrar, usando métodos o algoritmos de aprendizaje, una función *clasificadora* $\gamma : \mathscr{X} \to \mathscr{C}$, que mapee documentos a clases, que satisfaga que $D \subset \gamma$.

Clasificación V

El aprendizaje que toma parte en la búsqueda de γ es llamado *aprendizaje supervisado* debido a que se necesita la ayuda de uno o varios expertos que creen el conjunto de entrenamiento D. Estos expertos son también quienes determinan el conjunto de clases en que se clasificarán los textos. Por ahora este trabajo se enfocará nos enfocaremos en problemas en que cada documento se clasifica dentro de una sola clase.

Naive Bayes I

Uno de los métodos más comunes de aprendizaje supervisado es el conocido como *Naive Bayes* (NB). Este es un método de aprendizaje probabilístico. La probabilidad de un documento d de pertenecer a una clase c se puede expresar como $P(c \mid d)$. La tarea del algoritmo es encontrar la mejor clase para cada documento d. Para ello NB establece que la clase más apropiada para un documento es la más probable, o sea

$$c_{map} = \arg\max_{c \in \mathscr{C}} P(c \mid d).$$

Naive Bayes II

La clase escogida para d se denota por c_{map} debido a que este método de clasificación, de acuerdo a la clase más probable para un documento dado, es conocido como *maximum a posteriori* (MAP).
Sin embargo la probabilidad condicional $P(c \mid d)$ es difícil de determinar.
Haciendo uso del *Teorema de Bayes* la probabilidad anterior puede ser expresada como

$$P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)}.$$

Naive Bayes III

El factor de normalización $P(d)$ es usualmente ignorado ya que no aporta información a la hora de buscar la clase más apropiada para un documento d, ya que este tiene el mismo efecto en todos los candidatos. Este cálculo puede ser simplificado si lo expresamos en términos de los términos en los documentos. Supongamos que $\{t_1, t_2, \ldots, t_n\}$ son los términos que aparecen en d. Entonces tenemos que

$$c_{map} = \arg\max_{c \in \mathscr{C}} P(c)P(d \mid c) = \arg\max_{c \in \mathscr{C}} P(c) \prod_{1 \leq k \leq n} P(t_k \mid c),$$

donde $P(t_k \mid c)$ es la probabilidad de que el término t_k aparezca en un documento de la clase c.

Naive Bayes IV

Podemos considerar $p(t_k \mid c)$ como una medida de qué tanto demuestra el término t_k que c es la clase correcta. El término $P(c)$ es conocido como probabilidad previa (*prior probability*) y en caso de que la información aportada por los términos no sea determinante en la selección podemos siempre escoger la clase con mayor valor de $P(c)$. Para simplificar aun más el cómputo se pueden sustituir los valores anteriores por sus logaritmos. Esto reducirá el costo de hacer los cálculos y además los errores aritméticos, dado que la multiplicación se transforma en suma. La clase seleccionada sería entonces

$$c_{map} = \arg\max_{c \in \mathscr{C}} \left(\log(P(c)) + \sum_{1 \leq k \leq n} \log(P(t_k \mid c)) \right).$$

Naive Bayes V

Solo queda ver como se estiman los parámetros $P(c)$ y $P(t_k \mid c)$, dado que los valores reales no son posibles de calcular. Para la probabilidad previa se puede contar la frecuencia relativa de cada clase en D:

$$P(c) = \frac{N_c}{N},$$

donde N_c es el número de documentos en la clase c y N es el número total de documentos. Se procede de manera similar para la probabilidad específica de una palabra en una clase

$$P(t_k \mid c) = \frac{T_{c,t_k}}{T_c},$$

Naive Bayes VI

donde T_{c,t_k} indica la cantidad de veces que ocurre la palabra t_k en todos los documentos de la clase c y T_c es la cantidad total de palabras (contando repeticiones) en toda la clase c.

Naive Bayes VII

Sin embargo, aún tenemos un problema con estas fórmulas y es que estamos asignando probabilidad cero a todos las clases que no contengan a todas las palabras del documento a clasificar. Para evitar esto adicionamos por defecto una unidad a cada contador lo cual es conocido como *Laplace smoothing*

$$P(t \mid c) = \frac{T_{c,t} + 1}{T_c + |V|},$$

donde $|V|$ es el número total de téminos en el vocabulario.
Es importante destacar que en este método se está obviando la posición de las palabras. Se presentan aquí los Algoritmos 3 y 4 para entrenar y clasificar usando BN que fueron tomados de [**?**, Figure 13.2].

Naive Bayes VIII

Algoritmo 3 TrainMultinomial

Parámetros de entrada: Conjunto de clases $\mathcal{C}$ y conjunto de entrenamiento D.

1: $V \leftarrow ExtractVocabulary(D)$
2: $N \leftarrow CountDocs(D)$
3: **for** $c \in \mathcal{C}$ **do**
4: $\quad N_c \leftarrow CountDocsInClass(D, c)$
5: $\quad prior[c] \leftarrow N_c/N$
6: $\quad text_c \leftarrow ConcatenateTextOfAllDocsInClass(D, c)$
7: $\quad$ **for** $t \in V$ **do**
8: $\quad\quad T_{ct} \leftarrow CountTokensOfTerm(text_c, t)$
9: $\quad$ **for** $t \in V$ **do**
10: $\quad\quad condprob[t][c] \leftarrow \frac{T_{c,t}+1}{\sum_{t'}(T_{c,t'}+1)}$
11: **return** $V, prior, condprob$

Algoritmo para entrenar Naive Bayes tomado de [1, Figura 13.2]

Naive Bayes IX

Algoritmo 4 ApplyMultinomialNB

Parámetros de entrada: $\mathcal{C}, V$, $prior$, $condprob$, d

1: $W \leftarrow ExtractTokensFromDoc(V, d)$
2: **for** $c \in \mathcal{C}$ **do**
3: $\quad score[c] \leftarrow \log prior[c]$
4: $\quad$ **for** $t \in W$ **do**
5: $\quad\quad score[c] += \log condprob[t][c]$
6: **return** $\arg\max_{c \in \mathcal{C}}(score[c])$

Algoritmo para aplicar Naive Bayes tomado de [1, Figura 13.2]

Naive Bayes X

Podemos deducir de los algoritmos que la complejidad de ambos es linear en el tiempo que toma escanear la información. Dado que esto hay que hacerlo al menos una vez, se puede decir que este método tiene complejidad temporal óptima. Dicha eficiencia hace que NB sea un método de clasificación tan usado.

Feature Selection

K Nearest Neighbor

Medidas de evaluación

Ventajas

Desventajas

Aplicaciones en la Recuperación de la Información

Otros ejemplos de aplicación