Sistemas de Recuperación de Información

Aplicaciones del agrupamiento y de la clasificacin en la recuperación de información en la Web

---

Autores

Laura Victoria Riera Pérez

Marcos Manuel Tirador del Riego

Índice general

1. Agrupamiento

- Aprendizaje no supervisado
- Problema que resuelve

1.1. Algunas definiciones

- Flat clustering
- Hierarchical clustering
- Hard clustering
- Soft clustering
- Hipótesis de agrupamiento
- Cardinalidad

1.2. Flat clustering

- Medida de similitud:
- Medidas de evaluacion:
  - Criterio interno de calidad
  - Criterio interno de calidad
  - Pureza
  - Índice de frontera?
  - Medida F
- Algoritmos:
  - K-means
  - EM (generalización de K-means)

1.3. Hierarchical clustering

- Hierarchical agglomerative clustering
- Medidas de similitud:
  - Single link clustering
  - Complete link clustering
  - Centroid clustering
- Evaluación de calidad:
  - Group average link
  - Método de Ward
- Divisive clustering
- Cluster labeling
- Algoritmos:
  - Algoritmo HAC
  - Divisive Clustering

1.4. Aplicaciones a la RI

- Search result clustering
- Scatter-Gather
- Collection clustering
- Language modeling
- Cluster-based retrieval

1.5. Ventajas

1.6. Desventajas

2. Clasificación

El problema de clasificación en sentido general consiste en determinar dentro un conjunto de clases a cuál de ellas pertenece un objeto dado. En el marco de este documento estamos interesados en 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.

Como alternativa podría pensarse el uso de *reglas* para determinar automáticamente si un texto pertenece o no a una clase determinada de documentos.

Ilustremos un ejemplo de regla aplicada automáticamente. Supongamos que un usuario necesita hacer una consulta en una página de noticias, por ejemplo, necesita tener actualidad sobre noticias relacionadas con las finanzas para tomar decisiones en su negocio. Al usuario entonces le podría interesar que de hacer una consulta en el sistema dicha consulta se mantenga ejecutando y le provea periódicamente las noticias relativas a finanzas.

A este tipo de consultas se le denomina 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 query 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.

Existe, sin embargo, un enfoque adicional a los dos anteriores mencionados. Nos referimos al 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.

Introduciremos a continuación la definición forma del problema de clasificación de textos, en el contexto del Aprendizaje de Máquinas.

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

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. Denotaremos el método de aprendizaje supervisado descrito por Γ, el cual actúa como una función que mapea un conjunto de datos de entrenamiento en una función clasificadora, osea que $\Gamma(D) = \gamma$.

La definición dada en 1 implica que cada documento pertenece a una sola clase. Pero existe otro tipo de problemas que permiten que un documento pertenezca a más de una clase. Por ahora enfocaremos nuestra atención en el tipo de una clase.

2.1. Problemas de la Recuperación de la Información

El aprendizaje de máquinas es ampliamente usado en la actualidad para resolver una variedad de problemas de muchas esferas. La clasificación, de conjunto con la regresión, es una de las grandes tareas que tiene el aprendizaje de máquinas supervisado. Algunos problemas interesantes que resuelven la clasificación se encuentran dentro del marco de los Sistemas de Recuperación de la información. Anteriormente presentbamos uno de ellos, que es la recuperación de consultas permanentes.

La organización del correo personal es otros de los ejemplos relacionados con RI que pueden ser resueltos mediante clasificación. En muchas ocasiones una persona tiene un número grande de correos en el buzón de entrada. A la hora de revisarlo sería deseable que pudiera dirigirse directamente a aquellos que son de interés para él. Para ello la organización automática del correo en carpetas podría ser una ventaja invaluable. Un ejemplo particular sería la carpeta de correo spam.

Otro problema que se pudiera resolver sería la clasificación de valoraciones sobre algo en positivas o negativas. Esto tendría numerosas aplicaciones prácticas como pudiera ser la selección de una película. Un usuario podría revisar la cantidad de comentarios negativos, antes de lanzarse a disponer de dos horas de su tiempo viendo un material audiovisual que no resultará muy placentero.

El uso más evidente que tiene la clasificación el la RI es la clasificación de documentos en tópicos. Esta clasificación facilitaría la implementación de un motor de búsqueda vertical, que permita al usuario restringir su búsqueda al tema deseado. Un SRI con semejante característica permite hacer una búsqueda más precisa en las consultas más especializadas por el usuario.

En RI es muy útil contar con un índice de los contenidos almacenados. En el proceso de creación del mencionado índice la clasificación puede jugar un importante papel. Por citar algunos ejemplos puede usarse para detectar el lenguaje de un documento, la segmentación y capitalización de las palabras y el codificado del documento.

2.2. Naive Bayes

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

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

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

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

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

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

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

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

Solo nos queda ver como estimamos los parámetros $P(c)$ y $P(t_k \mid c)$, dado que los valores reales no son posibles de calcular. Para la probabilidad previa podemos 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 numero total de documentos. Procedemos de manera similar para la probabilidad específica de una palabra en una clase

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

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. Si 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 estamos obviando la posición de las palabras. Presentamos aquí los algoritmos para entrenar y clasificar usando BN que fueron textualmente copiados de 2009 Manning C. D., Introduction to Information Retrieval, página 260. [No recuerdo como citar esto correctamente. Also recordar que agregue unos paquetes arriba que no se si se agragan aqui o en otro de los .tex].

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.

2.3. Feature Selection

Un término con ruido (*noise feature*) es aquel que al pertenecer a la representación de los documentos, provoca un aumento del error de clasificación de los datos. Por ejemplo, supongamos que tenemos una palabra que ocurre rara vez, pero que en el conjunto de entrenamiento ocurre siempre en la misma clase c. Entonces al clasificar un documento nuevo que contiene esta palabra, la misma provocará que el clasificador se incline en cierta medida por seleccionar esta a c como respuesta. Sin embargo, dado que la ocurrencia de esta palabra

Algorithm 1 TrainMultinomial

Require: Set of classes $\mathcal{C}$ and training set D.

$V \leftarrow ExtractVocabulary(D)$
$N \leftarrow CountDocs(D)$
for $c \in \mathcal{C}$ **do**
  $N_c \leftarrow CountDocsInClass(D, c)$
  $prior[c] \leftarrow N_c/N$
  $text_c \leftarrow ConcatenateTextOfAllDocsInClass(D, c)$
  for $t \in V$ **do**
    $T_{ct} \leftarrow CountTokensOfTerm(text_c, t)$
  for $t \in V$ **do**
    $condprob[t][c] \leftarrow \frac{T_{c,t}+1}{\sum_{t'}(T_{c,t'}+1)}$
return $V, prior, condprob$

Algorithm 2 ApplyMultinomialNB

Require: $\mathcal{C}, V$, $prior$, $condprob$, d

$W \leftarrow ExtractTokensFromDoc(V, d)$
for $c \in \mathcal{C}$ **do**
  $score[c] \leftarrow \log prior[c]$
  for $t \in W$ **do**
    $score[c] += \log condprob[t][c]$
return $\arg\max_{c \in \mathcal{C}}(score[c])$

únicamente en c es accidental, claramente no aporta información suficiente para la clasificación y por tanto, al considerar lo contrario, aumenta el error.

Este es uno de los propósitos que tiene la selección de términos (*feature selection* (FS)). Esta consiste en reducir el vocabulario, considerado en la clasificación de textos solo un subconjunto del que aparece en el conjunto de entrenamiento. Nótese que al disminuir el tamaño del vocabulario aumenta la eficiencia de los métodos de entrenamiento y clasificación (aunque no es el caso de NB).

Selección de términos prefiere un clasificador más simple antes que uno más complejo. Esto es útil cuando el conjunto de entrenamiento no es muy grande.

En FS usualmente fijamos una cantidad k de vocablos por cada clase c, que serán los usados por el clasificador. Para seleccionar los k términos deseados establecemos un ranking entre los términos de la clase, haciendo uso de una función de medida de utilidad $A(t, c)$, y nos quedamos con los k mejor posicionados. El algoritmo básico consiste en para cada clase c iterar por todos los términos del vocabulario y computar su medida de utilidad para la clase; para finalmente ordenar los resultados y devolver una lista con los k mejores.

Presentaremos a continuación tres de los métodos de calcular $A(t, c)$ más comunes.

- **Información Manual.**
  Computar $A(t, c)$ como el valor esperado de información mutua (*Mutual Information* (MI)), nos da una medida de cuánta información aporta, la pre-

sencia en c de un término dado, a tomar la decisión correcta de clasificación de un documento. Lo definimos como [Hay que poner aqui que esto se cogio del libro, pagina 272]

$$I(U_t; C_t) = \sum_{e_t \in \{1,0\}} \sum_{e_c \in \{1,0\}} P(U_t = e_t, C_t = e_c) \log_2 \frac{P(U_t = e_t, C_t = e_c)}{P(U_t = e_t)P(C_t = e_c)},$$

donde U_t es una variable aleatoria que toma valor $e_t = 1$ si el documento contiene el término t y $e_t = 0$ en otro caso, y C es otra variable aleatoria que toma valor $e_c = 1$ si el documento está en la clase c y $e_c = 0$ en otro caso.

MI mide cuánta información un término contiene acerca de una clase. Por tanto, mantener los términos que están cargados de información, y eliminar los que no, contribuye a reducir el ruido y mejorar la precisión del clasificador.

- **Selección Chi cuadrado χ^2.**

  En estadística se dice que dos eventos son independientes si el resultado de uno no afecta al resultado del otro. Esto se puede escribir formalmente como $P(AB) = P(A)P(B)$. En estadística el test χ^2 se usa para medir el grado de independencia de dos eventos. En FS podemos entonces considerar aplicar este test asumiendo como eventos la ocurrencia de los términos y la ocurrencia de las clases. Esto es Esto tambien hay que poner de donde lo cogi (pag 275)

$$\chi^2(D, t, c) = \sum_{e_t \in \{1,0\}} \sum_{e_c \in \{1,0\}} \frac{(N_{e_t e_c} - E_{e_t e_c})^2}{E_{e_t e_c}},$$

  donde N es la frecuencia según D, E es la frecuencia esperada y e_t y e_c se definen como en la medida anterior.

- **Selección basada en frecuencia**.

  Esta medida consiste en priorizar los términos que son más comunes en la clase. Puede ser calculada de dos formas diferentes. La primera es cantidad de repeticiones de un término en los documentos de una clase, conocida como frecuencia en colección. La otra es frecuencia de documentos, y se calcula como la cantidad de documentos en la clase que contienen al término en cuestión.

  Cuando son seleccionados varios miles de términos, entonces esta medida es bastante buena. Esta es preferible a otros métodos más complejos cuando se aceptan soluciones subóptimas.

2.4. K Nearest Neighbor

En el algoritmo de Naive Bayes representábamos los documentos como vectores booleanos de términos. Luego vimos que hay términos que no eran relevantes y que aportaban ruido, y lo solucionamos seleccionando para el clasificador solamente un subconjunto de todos los términos. Aún así estamos clasificando la

relevancia de cada término de manera binaria en relevante o no relevante (que aporta ruido).

El método que presentamos en esta sección, así como otros similares, asignan a cada término cierto valor de importancia relativa al documento en que aparece. Para esto se cambia la representación de los documentos a vectores de $\mathbb{R}^{|V|}$, donde a cada componente corresponde cierto peso que se le asigna al término correspondiente a esta. Entonces, el espacio de documentos $\mathcal{X}$ (dominio de γ) es $\mathbb{R}^{|V|}$. A esta forma de representación de documentos se le conoce como modelo de espacio de vectores. La hipótesis básica para usar el modelo de espacio de vectores es la siguiente citar adecuadamente : pagina 289

Hipótesis de contigüidad: Documentos en la misma clase forman una región contigua y regiones de diferentes clases no se superponen.

Las decisiones de muchos clasificadores basados en espacio de vectores dependen de una noción de distancia. Pueden ser usadas por ejemplo similitud basado en el coseno (del ángulo formado entre los vectores) o distancia Euclideana. Por lo general no hay mucha diferencia entre usar una u otra de estas distancias.

La tarea de la clasificación en el modelo de espacio de vectores es determinar las fronteras entre los documentos pertenecientes a una u otra clase. Estas últimas son llamadas fronteras de decisión ya que dividen el espacio en diferentes poliedros, tales que si un documento pertenece a uno determinado, automáticamente sabemos de qué clase es.

En K Nearest Neighbor la frontera de decisión se determina localmente. En este asignamos cada documento a la misma clase que la mayoría de los k puntos más cercanos al documento. Basado en la hipótsis de contigüidad esperamos que el documento d pertenezca a la misma clase que aquellos más cercanos a él.

En kNN para subdividir el espacio de documentos en regiones, dado un $k \in \mathbb{N}$ fijo, consideramos cada región como el conjunto de puntos para los cuales los k puntos más cercanos son los mismos. Estas regiones son poliedros convexos. Luego para cada una de estas regiones existe una clase a la que pertenecen todos sus puntos, que es aquella a la que pertenecen la mayoría de los documentos, ya clasificados, que están dentro de la región. En caso de que haya empate, la decisión de a que clase asignar a un nuevo documento que pertenece a esta región del espacio, es tomada aleatoriamente entre las clases empatadas.

El parámetro k es usualmente seleccionado basado en el conocimiento que se posee sobre los problemas de clasificación similares al que se tiene. Otra forma de seleccionar k es usando conjuntos de documentos de prueba (ya clasificados) para ver que valor de k producen mejores resultados.

También hay variantes del método donde lo que se hace es calcular una similitud entre el documento d a clasificar y cada uno de los k más cercanos, usando, por ejemplo, el coseno entre los vectores. Luego se hace un ranking entre las clases a las que pertenecen cada uno de los k puntos. Para ello se calcula para cada c, la suma de la similitud entre d y cada uno de los puntos que pertenecen a c y a la vez están entre los k mencionados. La siguiente función *score* produce

el resultado deseado

$$score(c,d) = \sum_{d' \in S_k(d)} I_c(d') \cos(\overrightarrow{v}(d'), \overrightarrow{v}(d)),$$

donde $S_k(d)$ es el conjunto de los k puntos más cercanos a d e $I_c(d')$ es 1 o 0 en dependencia de si d' pertenece a la clase c o no. Finalmente se selecciona para d la clase c que más alto aparezca en el ranking. En ocasiones esta variante presenta mayor exactitud que la anterior.

2.5. Evaluación de la clasificación

Una alta exactitud en la clasificación de los datos de entrenamiento no necesariamente se traduce en resultados correctos en los nuevos documentos introducidos. Puede sucede que el sistema resulte sobreentrenado (*over-fitting*) o subentrenado (*under-fitting*). El primero de los casos ocurre cuando el modelo se encuentra con muchos datos. En este caso el sistema aprende de los ruidos y de las entradas inexactas lo que provoca que el clasificador aprenda incorrectamente a clasificar los elementos de alguna clase. Por otro lado under-fitting ocurre cuando el sistema tiene información muy vaga sobre la frontera entre clases lo que provoca cierta aleatoriedad en la clasificación.

Para evaluar la labor de un clasificador se debe emplear un conjunto de documentos diferentes al conjunto entrenante el cual se llama conjunto de prueba. Este debe estar en todo momento aislado del conjunto de entrenamiento. Si se juntan el clasificador puede aprender en su entrenamiento a reconocer los documentos del conjunto de prueba, y luego al evaluar el mismo, el desempeño del sistema será mucho más alto que el resultado real que tenga cuando se ponga en uso. Usualmente los datos que se tienen se dividen entre el conjunto entrenante y de prueba a razón de $4:1$.

Las siguientes tres métricas pueden ser utilizadas para evaluar clasificadores de dos clases (c y $\tilde{c}$):

- **Precisión**

$$Precision = \frac{N\acute{u}mero\ de\ documentos\ correctamente\ identificados\ como\ c}{N\acute{u}mero\ de\ documentos\ identificados\ como\ c}$$

- **Recobrado**

$$recobrado = \frac{N\acute{u}mero\ de\ documentos\ correctamente\ identificados\ como\ c}{N\acute{u}mero\ de\ documentos\ que\ pertenecen\ a\ c}$$

- **Medida** F

$$F = 2 \times \frac{Precision \times Recobrado}{Precision + Recobrado}$$

El mismo resultado se puede ver usando la matriz de contingencia. Una matriz de contingencia muestra de todas las pruebas realizadas con el clasificador, cuántas resultaron en verdadero positivo, falso positivo, falso negativo y verdadero negativo.

En el caso que la cantidad de clases sea mayor que dos, podemos computar promedios entre las métricas anteriores viendo el clasificador como uno de dos clases aplicado a cada clase c (y su complemento). Existen dos promedios que son los más usados para esto. Macropromedio calcula simplemente la media entre las medidas anteriores para cada clasificador de dos clases. Microaverage primero adiciona las matrices de contingencia correspondientes a diferentes clases y luego aplica una de las métricas anteriores sobre la matriz resultante.

NOTA:

- Ver lo de cuando estan hablando de clasificacion en dos clases y cuando no.
- Podría insertarse una foto de la matriz de contingencia(pagina 16 de classification.pdf)
- Ver si agrego algo de clasificacion any-of
- agregar el código del algoritmo de KNN.
- Ver lo de statistical classification que no me queda claro si toda lo que he escrito entra dentro de eso o no.
- citas correctas

- Aprendizaje supervisado
- Problema que resuelve

- Rule-based classification
- Statistical classification
- Feature selection
- Medidas de evaluación:
  - Fitting
  - Precisión
  - Recobrado
  - Medida F (balanceada)
  - Classification accuracy
- Algoritmos:
  - Naive Bayes
  - K-Nearest Neighbours

2.6. Aplicaciones a la RI

- Standing queries
- Spam filtering

2.7. Ventajas

2.8. Desventajas

3. Agrupamiento vs. Clasificación

4. Ejemplos de aplicacin