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.

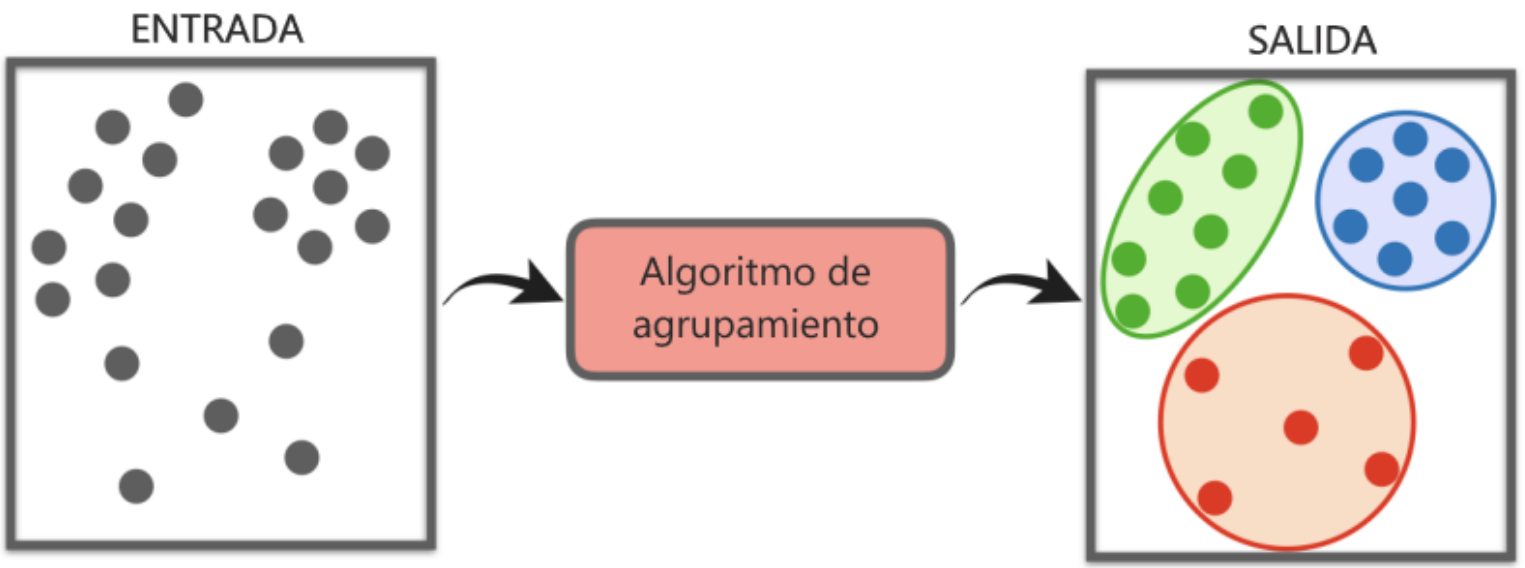
Noviembre, 2022

Índice general I

Índice general II

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 tambiÃľn sean relevantes.

Clasificación de los algoritmos de agrupamiento

Según el tipo de estructura impuesta sobre los datos:

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

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.

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

Se ha utilizado en la recuperaciÃșn de informaciÃșn para muchos propÃșsitos diferentes:

- expansiÃșn de consultas
- agrupaciÃșn e indexaciÃșn de documentos
- 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.

Clasificación II

Formas que requieren un alto esfuerzo humano:

- Manualmente.
- Uso de reglas (relacionado con las standing queries): Una regla captura una cierta combinación de palabras claves que identifican una clase.

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

- 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.
- 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.

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.

Naive Bayes II

La probabilidad de un documento d de pertenecer a una clase c se puede expresar como $P(c \mid d)$. 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 III

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 IV

- 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 función 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 V

Podemos considerar $p(t_k \mid c)$ como una medida de qué tanto demuestra el término t_k que c es la clase correcta.

Naive Bayes VI

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 VII

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.

Naive Bayes VIII

Se procede de manera similar para la probabilidad específica de una palabra en una clase

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

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 IX

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.

Naive Bayes X

Algoritmo 3 TrainMultinomial

Parámetros de entrada: Conjunto de clases $\mathcal{C}$ y conjunto de entrenamiento 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 $V, prior, condprob$

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

Naive Bayes XI

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 XII

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 I

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.

Feature Selection II

La selección de términos 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).

Feature Selection III

- En FS usualmente se fija una cantidad k de vocablos por cada clase c, que serán los usados por el clasificador.
- Para seleccionar los k términos deseados se establece un ranking entre los términos de la clase, haciendo uso de una función de medida de utilidad $A(t, c)$, y luego se escogen los k mejor posicionados.

Feature Selection IV

Se presenta 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)), da una medida de **cuánta información aporta, la presencia en c de un término dado, a tomar la decisión correcta de clasificación de un documento.**

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

  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.

Feature Selection V

- **Selección Chi cuadrado χ^2.**

  El test χ^2 se usa para medir el grado de independencia de los eventos: **ocurrencia de los términos y ocurrencia de las clases.**

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

  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.

Feature Selection VI

- **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.**

K Nearest Neighbor I

El método que será presentado 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 $\mathscr{X}$ (dominio de γ) es $\mathbb{R}^{|V|}$. A esta forma de representación de documentos se le conoce como **modelo de espacio de vectores**.

K Nearest Neighbor II

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

K Nearest Neighbor III

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.

K Nearest Neighbor IV

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.

K Nearest Neighbor V

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

K Nearest Neighbor VI

Algoritmo 5 Train-kNN

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

1: $D' \leftarrow PreProcess(D)$
2: $k \leftarrow Selectk(\mathcal{C}, D')$
3: **return** D', k

Algoritmo para entrenar KNN tomado de [1, Figura 14.7]

K Nearest Neighbor VII

Algoritmo 6 Apply-kNN

Parámetros de entrada: $\mathcal{C}$, D', k, d

1: $S_k \leftarrow ComputeNearestNeighbors(D', k, d)$
2: **for** $c_j \in \mathcal{C}$ **do**
3: $\quad p_j \leftarrow |S_k \cap c_j|/k$
4: **return** $\arg\max_j p_j$

Algoritmo para aplicar KNN tomado de [1, Figura 14.7]

K Nearest Neighbor VIII

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 qué valor de k producen mejores resultados.

Medidas de evaluación I

Una alta exactitud en la clasificación de los datos de entrenamiento no necesariamente se traduce en resultados correctos en los nuevos documentos introducidos. Puede suceder que el sistema resulte sobreentrenado (*over-fitting*) o subentrenado (*under-fitting*).

Medidas de evaluación II

El primero de estos ocurre cuando un sistema se entrena demasiado o se entrena con datos extraños. En este caso el algoritmo de aprendizaje puede quedar ajustado a unas características muy específicas de los datos de entrenamiento que no tienen relación causal con la función objetivo. Entonces el clasificador puede aprende 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.

Medidas de evaluación III

Para evaluar la labor de un clasificador se debe emplear un conjunto de documentos diferente al conjunto entrenante el cual se llama conjunto de prueba. Este debe estar en todo momento aislado del conjunto de entrenamiento. Usualmente los datos que se tienen se dividen entre el conjunto entrenante y de prueba a razón de $4:1$.

Medidas de evaluación IV

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

Medidas de evaluación V

- **Precisión**

$$Precision = \frac{tp}{tp + fp},$$

donde tp es el número de documentos clasificados correctamente como de c y $tp + fp$ es el total de documentos clasificados como c.

Medidas de evaluación VI

- **Recobrado**

$$Recobrado = \frac{tp}{tp + fn},$$

donde tp es el número de documentos correctamente identificados como c y $tp + fn$ es el número de documentos que pertenecen a c.

Medidas de evaluación VII

- **Medida** F

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

Medidas de evaluación VIII

Podemos ver las clasificaciones anteriores en la matriz de contingencia.

		Gold standard	
		Positive	**Negative**
Actual classification:	**positive**	True positive (tp)	False positive (fp)
	Negative	False negative (fn)	True negative (tn)

Medidas de evaluación IX

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. Micropromedio primero adiciona las matrices de contingencia correspondientes a diferentes clases y luego aplica una de las métricas anteriores sobre la matriz resultante.

Ventajas I

Ventajas:

- Al usar el aprendizaje supervisado se puede tener una idea exacta sobre las clases de los documentos, dado que estas se crean en base a características explícitas de los mismos.
- La salida suele ser más precisa que en el aprendizaje supervisado
- El proceso es fácil de comprender ya que las acciones de la máquina se reconocen con precisión.
- Es más facil de trabajar con el aprendizaje supervisado que con el no supervisado.
- Se evalúa y recibe retroalimentación para comprobar la correctitud.

Desventajas I

Desventajas:

- Se necesita un especialista que determine cuáles serán las clases en que se desea clasificar.
- Se requiere de un conjunto de datos entrenantes.
- Por lo general no se resuelven tareas tan complejas como en el aprendizaje supervisado.
- Los algoritmos solo funcionan dentro de las restricciones que se le han impuestos por lo que no proporcionan soluciones creativas.
- Suelen requerir mucho tiempo de entrenamiento.
- Puede predecir la salida incorrecta si los datos de prueba son diferentes de los datos de entrenamiento.

Aplicaciones en la Recuperación de la Información I

- Recuperación de consultas permanentes.

Aplicaciones en la Recuperación de la Información II

- La organización del correo personal: Es otros de los ejemplos relacionados con la Web 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.

Aplicaciones en la Recuperación de la Información III

- 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. Los sistema que se especializan en este tipo de sugerencias de contenido, son llamados usualmente **sistemas de recomendación**.

Aplicaciones en la Recuperación de la Información IV

- 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.

Aplicaciones en la Recuperación de la Información V

- Creación de índices de los contenidos almacenados en un SRI: En el proceso de creación de los mencionados índices 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.

Otros ejemplos de aplicación I

- La bioinformática: Con el uso de sus algoritmos se permite automatizar la búsqueda de patrones en conjuntos de datos, que puedan ayudar a entender diferentes procesos biológicos que se manifiestan en los mismos. Debido a los grandes volúmenes de datos que se tienen ha crecido mucho en los últimos años la aplicación de la clasificación en esta disciplina.

Otros ejemplos de aplicación II

- Predicción de gen. Esta consiste en determinar que fracciones de una secuencia de ADN codifican proteínas.
- Investigación de comunidades de microbios en nichos ecológicos. Así se puede obtener información taxonómica y metabólica de las comunidades estudiadas.
- Proteómica
- Microarrays
- La biología de sistemas.

Otros ejemplos de aplicación III

- Reconocimiento de algunos rasgos distintivos de una persona, como pueden ser, reconocimiento de rostros, huellas o voz. Este último lo tenemos al alcance de nuestras manos en asistentes virtuales como Siri o Google. Estas aplicaciones se entrenan para reconocer características distintivas de tu voz, mediante un proceso de aprendizaje supervisado, para luego poder diferencias cúando eres tú quien les está hablando.

Otros ejemplos de aplicación IV

- El mercado financiero: Se puede emplear en mejorar el mercado digital, las ventas en línea, identificar el valor de vida útil de un cliente, la tasa de abandono, recomendaciones de productos y análisis de impacto de campañas de mercado.

Otros ejemplos de aplicación V

- En la seguridad informática: acciones como detección de virus, enlaces maliciosos y fraude.
- Internet de las cosas (IoT).
- Los motores de recomendación
- La fijación de precios dinámicos de productos.