

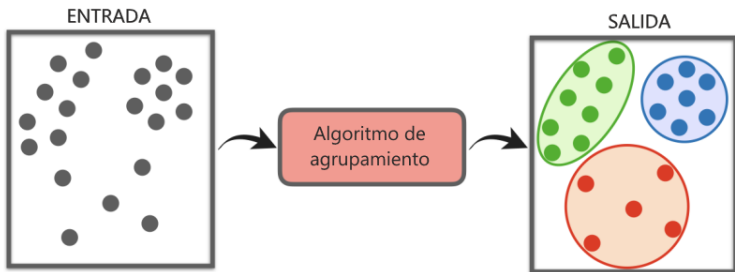
## An illustration showing two people, a man and a woman, reviewing a large document titled 'ANALYSIS'. The man is holding a large magnifying glass over the document, which features various charts and graphs. The woman is standing next to him, also looking at the document. The document is held up by a large green clip. The background is white with some green foliage and a grey cloud.

Marcos Manuel Tirador del Riego  
Laura Victoria Riera Pérez

Noviembre, 2022

# Agrupamiento I

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



- Es la forma más común de aprendizaje no supervisado.
- Los grupos formados deben tener un alto grado de asociación 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.

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

La hipótesis establece que si hay un documento de un grupo que es relevante a una solicitud de búsqueda, entonces es probable que otros documentos del mismo clúster también sean relevantes.

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 clústeres sin ninguna estructura explícita que los relacione entre sí.

El algoritmo más importante de este tipo de agrupamiento es el K-means.

## Agrupamiento jerárquico

El agrupamiento jerárquico produce una jerarquía, una estructura que es más informativa que el conjunto no estructurado de clústeres 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 clúster único 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 jerárquico aglomerativo o HAC por sus siglas en inglés.

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





# Algoritmo HAC I

Dado un conjunto de  $N$  elementos a agrupar, el proceso básico del agrupamiento jerárquico aglomerativo es:

- 1 Se comienza con  $N$  clústeres, resultado de asignar cada elemento al suyo propio. Se computa la matriz  $C$  de similitud de  $N \times N$ .
- 2 Se halla la similitud entre los pares de clústeres con la medida deseada.
- 3 Se toma el par más similar de clústeres y se combinan en un único clúster.
- 4 Se calculan las similitudes entre el nuevo clúster y cada uno de los clústeres antiguos.
- 5 Se repiten los pasos 3 y 4 hasta que todos los elementos estén agrupados en un solo grupo de tamaño  $N$ .

---

**Algoritmo 2 HAC**

1: **for**  $n \leftarrow 1$  **to**  $N$  **do**2:   **for**  $i \leftarrow 1$  **to**  $N$  **do**3:  $C[n][i] \leftarrow SIM(d_n, d_i)$ 

4:  $I[n] \leftarrow 1$  (lleva los clústeres activos)

5:  $A \leftarrow \square$  (secuencia de mezclas aplicadas en el agrupamiento)

6: **for**  $k \leftarrow 1$  **to**  $N - 1$  **do**7:  $\langle i, m \rangle \leftarrow \arg \max_{\langle i, m \rangle: i \neq \wedge I[i]=1 \wedge I[m]=1} C[i][m]$ 

8:  $A.APPEND(< i, m >)$  (almacena la mezcla)

9:   **for**  $j \leftarrow 1$  **to**  $N$  **do**10:  $C[i][j] \leftarrow SIM(i, m, j)$ 11:  $C[j][i] \leftarrow SIM(i, m, j)$ 

12:  $I[m] \leftarrow 0$  (desactiva el clúster)

```
13: return A
```

## Agrupamiento jerárquico divisivo

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

Se beneficia de la información completa sobre la distribución global al tomar decisiones de partición 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 útil 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 aplicación en disímiles campos, cuenta el apoyo de una serie de paquetes de software, a menudo disponibles en la informática académica 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 recuperación de información para muchos propósitos diferentes:

- expansión de consultas
- agrupación e indexación de documentos
- visualización de resultados de búsqueda

Permiten mejorar interfaz y experiencia de usuario y proporcionar una mayor eficacia o eficiencia del sistema de búsqueda.

## Ejemplos de aplicación en la web II

- **Agrupamiento de resultados de búsqueda** (Search result clustering): La presentación predeterminada de los resultados de búsqueda (documentos devueltos en respuesta a una consulta) en la recuperación de información es una lista sencilla. Los usuarios escanean la lista de arriba a abajo hasta que encuentran la información que buscan.  
En su lugar, en la agrupación en clústeres de resultados de búsqueda los documentos similares aparecen juntos, siendo más fácil escanear algunos grupos coherentes que muchos documentos individuales. Esto es particularmente útil si un término de búsqueda tiene diferentes significados.







# 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 queries* 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 estadístico.

## 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  $\mathcal{X}$  el espacio de documentos y  $\mathcal{C} := \{c_i \mid c_i \subset \mathcal{X}, i \in \{1, 2, \dots, n\}\}$  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*  $\gamma: \mathcal{X} \rightarrow \mathcal{C}$ , que mapee documentos a clases, que satisfaga que  $D \subset \gamma$ .



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  $P(c | d)$ . NB establece que la clase más apropiada para un documento es la más probable, o sea

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

# Naive Bayes III

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  $P(c | d)$  es difícil de determinar. Haciendo uso del *Teorema de Bayes* la probabilidad anterior puede ser expresada como

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

## 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  $\{t_1, t_2, \dots, t_n\}$  son los términos que aparecen en  $d$ . Entonces tenemos que

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

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

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

# Naive Bayes V

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

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

Solo queda ver como se estiman los parámetros  $P(c)$  y  $P(t_k | c)$ , 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

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

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  $c$ .

Sin embargo, aún tenemos un problema con estas fórmulas y es que estamos asignando probabilidad cero a todas 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*

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

Es importante destacar que en este método se está obviando la posición de las palabras.

# Naive Bayes IX

---

**Algoritmo 3** TrainMultinomial

---

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

```

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

```

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



---

**Algoritmo 4** ApplyMultinomialNB1:  $W \leftarrow \text{ExtractTokensFromDoc}(V, d)$ 2; for  $c \in \mathcal{C}$  do

```

3:   $score[c] \leftarrow \log prior[c]$ 

```

4:   **for**  $t \in W$  **do**

```

5:      score[c] += log condprob[t][c]

```

6: **return**  $\arg \max_{c \in \mathcal{C}} (\text{score}[c])$ 

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

Podemos deducir de los algoritmos que la complejidad de ambos es lineal 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.

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, supóngase que se tiene 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 a  $c$  como respuesta. Sin embargo, dado que la ocurrencia de esta palabra únicamente en  $c$  es accidental, claramente no aporta información suficiente para la clasificación y por tanto, al considerar lo contrario, aumenta el error.

## Feature Selection II

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).

# Feature Selection III

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 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. 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.

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

# Feature Selection IV

- 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

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.

# Feature Selection VI

- **Selección Chi cuadrado  $\chi^2$ .**

El test  $\chi^2$  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 VII

- **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.

# 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  $\mathcal{X}$  (dominio de  $\gamma$ ) 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 hipótesis de contigüidad se espera que el documento  $d$  pertenezca a la misma clase que aquellos más cercanos a él.

## K Nearest Neighbor VI

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.

# K Nearest Neighbor VII

---

## 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 VIII

---

**Algoritmo 6** Apply-kNN

---

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

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

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

---

# K Nearest Neighbor IX

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.

## K Nearest Neighbor X

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

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

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.

# 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 aprender 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. 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.

# 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**

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

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 fácil 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 sistemas 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 diferenciar cuándo eres tú quien les está hablando.

- 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.