
Universidad Politécnica de Puebla

# AlVot con testores por clase como sistemas de conjuntos de apoyo

*Alumna:*
Dalia Rodríguez Salas

*Asesores:*
Dr. Jorge de la Calleja Mora
Dr. José Arturo Olvera López

28 de julio de 2014

# Índice general

**1. Introducción** **9**

**2. Marco Teórico** **13**
2.1. AlVot clásico . . . . . 13
2.2. Testor Clásico . . . . . 16
2.3. Testor por clase . . . . . 20
2.4. AdaBoost . . . . . 23
2.5. Trabajo relacionado . . . . . 25

**3. AlVot con un sistema de conjuntos de apoyo por clase** **29**
3.1. AlVot PC . . . . . 29
3.2. Algoritmo basado en AlVot PC . . . . . 33
3.3. Algoritmo basado en PC con AdaBoost como selector de testores por clase . . . . . 35
3.4. Criterios de comparación para la obtención de testores . . . . 37

**4. Pruebas y resultados** **41**
4.1. Cantidad de características . . . . . 42
4.2. Desempeño de clasificación . . . . . 44
4.2.1. Comparación con AlVot clásico. . . . . 44
4.2.2. Comparación con otros algoritmos de la literatura. . . 45

**5. Conclusiones y trabajo futuro** **47**

**A. Sistema que implementa los algoritmos propuestos** **49**

# Índice de figuras


1.1. (a)Matriz de aprendizaje de caracteres. (b)Características para describir los caracteres. . . . . 10
1.2. Representación de los objetos de la Figura 1.1 (a) utilizando únicamente las características que permiten diferenciar al primer objeto ($\Phi$) del resto. . . . . 10

2.1. Ejemplo para clasificar $O_{new}$ mediante un algoritmo A basado en AlVot Clásico para un problema de 3 clases y 4 conjuntos de apoyo. . . . . 17
2.2. Hipótesis de 4 clasificicadores débiles. . . . . 25
2.3. Hipótesis de los clasificadores de la Figura 2.2 seleccionadas por AdaBoost en la iteración $k = 1$, $k = 2$ y $k = 3$. . . . . 25
2.4. Hipótesis del clasificador fuerte contruido por AdaBoost a partir de los clasificadores débiles de la Figura 2.2. . . . . 25
2.5. Etapas en el esquema de selección de características por clase propuesto en [Pineda-Bautista et al., 2011] . . . . . 27
2.6. Ejemplo de *una clase contra todas* para un problema de clasificación supervisada de tres clases. . . . . 28
2.7. Proceso de clasificación propuesto en [Pineda-Bautista et al., 2011] utilizando selección de características por clase. . . . . 28

3.1. Ejemplo para clasificar $O_{new}$ mediante un algoritmo $A$ basado en AlVot PC para un problema de 3 clases con 4, 2 y 3 conjuntos de apoyo para la clase 1,2 y 3 respectivamente. . . . 32
3.2. Esquema del cálculo de las distribuciones normales al comparar valores de una característica numérica $x$ . . . . . 39
3.3. Ejemplo de los pasos para definir un criterio de comparación booleano para la la característica numérica $x_k$ . . . . . 39

4.1. Moda de la cantidad de características utilizadas por los conjuntos de apoyo en los algoritmos propuestos basados en AlVot PC, AlVot PC+ y su análago basado en AlVot clásico. . . . . 43
4.2. Promedio de los porcentajes de clasificación al realizar 5 veces validación cruzada con 10 folds con los algoritmos K-NN (con K=3 y K=5), C4.5, Naive Bayes y los propuestos basados en AlVot PC, AlVot PC+. . . . . 46

A.1. Pasos para definir un criterio de comparación booleano para la la característica numérica $x_k$. . . . . 50
A.2. Pasos para definir un criterio de comparación booleano para la la característica numérica $x_k$. . . . . 51

# Índice de cuadros

4.1. Descripción de los conjuntos de datos utilizados en las pruebas. 41
4.2. Promedio del porcenteje de clasificación al evaluar los algoritmos propuestos basados en AlVot PC, AlVot PC+ y su análago basado en AlVot clásico. . . . . 44
4.3. Promedio del porcenteje de clasificación al evaluar los algoritmos K-NN (con K=3 y K=5), C4.5, Naive Bayes y los propuestos basados en AlVot PC, AlVot PC+. . . . . 46

# Capítulo 1

# Introducción

El problema de categorizar objetos en clases previamente identificadas basándose en ejemplos de objetos de cada una de las clases, es conocido como *problema de clasificación supervisada* y los objetos de ejemplo son llamados *muestra de aprendizaje* (MA). El modelo de algoritmos de votación (AlVot) permite resolver dicho problema, como su nombre lo indica se basa en votaciones, donde cada voto considera un subconjunto de características que describe a los objetos. Cada voto es representado por un valor numérico que indica qué tan análogo o similar es el nuevo a objeto a los de MA. Los subconjuntos de características utilizados en las votaciones son conocidos como *sistema de conjuntos de apoyo*.

Las características que describen a los objetos son comúnmente seleccionadas de manera que permitan diferenciar a los objetos de MA, proceso conocido como *selección de características*. Determinar qué características deben ser utilizadas para describir a los objetos en un problema de clasificación supervisada es muy importante, la clasificación correcta de objetos nuevos depende de un buen conjunto de características [Pineda-Bautista et al., 2011]. Sin embargo, algunas de las características seleccionadas pueden no ser del todo necesarias para una clase específica, lo cual se ilustra con el Ejemplo 1.1 [Lazo-Cortés et al., 1998].

---

**Ejemplo 1.1.** Considerar como MA a una colección de caracteres descritos por una matriz de puntos de 5x5 tal como se muestra en la Figura 1.1 (a). Cada objeto pertenece a una clase diferente, por lo tanto hay 12 clases. Cada objeto (caracter) está descrito por 25 características denotadas como $x_i$, tal como se muestra en la figura 1.1 (b).

Las características mostradas en la Figura 1.1 (b), permiten diferenciar la

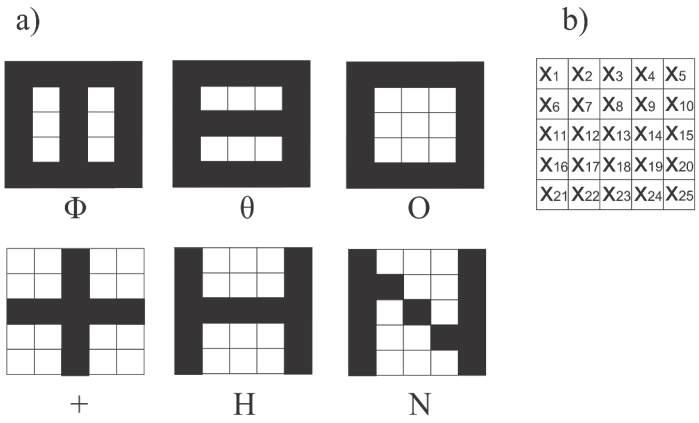


**Figura 1.1:** (a)Matriz de aprendizaje de caracteres. (b)Características para describir los caracteres.

clase a la que pertenece cada caracter, de esta manera pueden ser cosideradas como un buen conjunto de características, sin embargo, el primer objeto de la Figura 1.1 (a) que describe al caracter Φ, puede ser diferenciado del resto de objetos al seleccionar únicamente las características $x_3$,$x_6$ y $x_8$. En la Figura 1.2 se muestra dicha situación, dónde las celdas en color gris representan las características no seleccionadas.

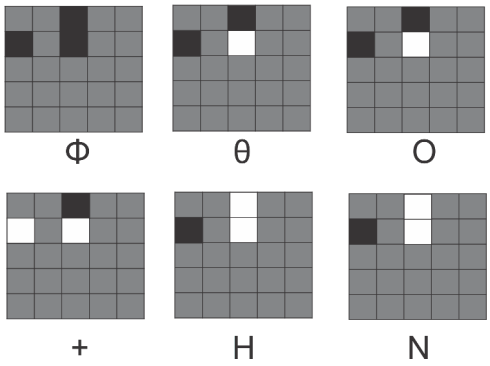


**Figura 1.2:** Representación de los objetos de la Figura 1.1 (a) utilizando únicamente las características que permiten diferenciar al primer objeto (Φ) del resto.

---

Del planteamiento de que algunas características seleccionadas pueden resultar innecesarias para identificar objetos de una clase específica surge el enfoque de *selección de características por clase*. Con el propósito de di-

ferenciar este enfoque, en adelante se denomina *selección de características clásica* al proceso de seleccionar aquellas caracaterísticas que diferencian a los objetos en MA entre todas las clases y no de cada una de ellas. En el contexto de la selección por clase, para resolver el problema de clasificación supervisada deben conocerse las características que diferencian a los objetos de cada una de las clases en MA.

AlVot -como se conoce hasta ahora- considera en su sistema de conjuntos de apoyo a subconjuntos de características seleccionadas de manera clásica. Este trabajo de tesis tiene como objetivo general, modificar AlVot de manera que permita considerar un sistema de conjuntos de apoyo para cada una de las clases. En adelante se denota AlVot clásico a aquel que considera un único sistema de conjuntos de apoyo y AlVot PC al que cosidera uno Por Clase.

En el contexto de este proyecto, un testor por clase está asociado a la selección de características por clase y un testor clásico al de selección clásica.

En su planteamiento más simple, un testor clásico asociado a MA, es un subconjunto de características no vacío que permite deferenciar la clase a la que pertenece cada uno de los objetos en MA. De manera análoga, un testor por clase es aquel que permite distinguir cada uno de los objetos en MA que pertenecen a una clase específica. MA puede tener asociado mas de un testor clásico o por clase, respectivamente.

Los algoritmos más eficientes para encontrar los testores asociados a MA, reciben como parámetro una matriz básica (MB) binaria. Esta es el resultado de comparar característica a característica los valores de todos los pares objetos en MA de distintas clases y eliminar filas redundantes. Para comparar dichos valores son necesarios *criterios de comparación*.

Los objetivos particulares de este trabajo de tesis son cuatro; el primero es definir los criterios de comparación entre valores de características para la obtención de testores, el segundo es desarrollar el sistema de cómputo que implemente un algoritmo basado en AlVot con los testores por clase como sistemas de conjuntos de apoyo, el tercero es implementar AdaBoost con el fin de seleccionar los testores por clase con mayor discriminanación en el alogoritmo propuesto y por último evaluar el desempeño de los algoritmos propuestos respecto a AlVot clásico y otros algoritmos de clasificación ampliamente utilizados en la literatura.

Los resultados experimentales muestran que el algoritmo propuesto basado en AlVot PC es significativamente más eficiente en la clasificación que AlVot clásico cuando los testores por clase son seleccionados mediante AdaBoost. En las pruebas realizadas también se muestran mejoras en la precisión respecto a otros algoritmos no basados en AlVot.

Este documento está organizado de la siguiente manera: en el capítulo 2

se plantea el marco teórico en el que se basa el algoritmo propuesto, además de los trabajos que se han realizado al respecto. En el capítulo 3 se define AlVot que utiliza un sistema de conjuntos de apoyo para cada clase, presentando en el capítulo 4 los resultados de evaluar y comparar el desempeño de clasificación del algoritmo desarrollado respecto a AlVot clásico y otros en el estado del arte. Por último en el capítulo 5 se muestran las conclusiones y trabajo futuro propuesto.

# Capítulo 2

# Marco Teórico

En este capítulo se presentan los conceptos teóricos necesarios para definir AlVot PC. Se comienza con un apartado que define AlVot clásico, para continuar con el concepto de testor y el algoritmo AdaBoost. Seguidamente, se incluye una sección del trabajo relacionado respecto a la selección de características por clase donde se adentra en el trabajo propuesto en [Pineda-Bautista et al., 2011], debido a su versatilidad para utilizar cualquier clasificador.

## 2.1. AlVot clásico

AlVot clásico [Zhuravlev and Nikiforov, 1971] permite resolver problemas de clasificación supervisada. Está basado en la idea de analogías parciales, esto es, un objeto puede parecerse a otro pero no en su totalidad, el parecido en algunas características permiten establecer ciertas propiedades.

Como se dijo anteriormente, en el enfoque clásico de AlVot se utilizan subconjuntos de características para comparar al objeto nuevo con los de MA, conocidos como conjuntos de apoyo. Para definir AlVot clásico, se presenta la siguiente notación:

- $A$. Algoritmo basado en AlVot clásico.
- $\Omega^A$. Sistema de conjuntos de apoyo de $A$.
- $n$. Número de características que describen a los objetos en MA.
- $R = \{x_1, x_2, ..., x_n\}$. Conjunto de características que describe a los objetos en MA.

- $\Omega_k = \{x_p, ..., x_s\}$. k-ésimo conjunto de apoyo, donde $1 \leq p \leq n$, $1 \leq s \leq n$, $\Omega_k \subseteq R$ y $\Omega_k \in \Omega^A$.
- $X_j$. *Vector de descripción total* del objeto $O_j$, de acuerdo a las características en $R$.
- $X_j^k$. *Vector de descripción parcial* del objeto $O_j$, de acuerdo a las caracteístias en el conjunto de apoyo $\Omega_k$.
- $x_p^j$. Valor de la característica $x_p \in R$ del objeto $O_j$.
- $m$. Número de clases en MA.
- $C = \{c_1, ..., c_m\}$. Conjunto de valores para las clases.
- $y_j$. Valor de la clase del objeto $O_j$, $y_j \in C$.

Para construir un algoritmo $A$ basado en AlVot Clásico, se deben determinar 6 etapas descritas a continuación:

1. **Sistema de conjuntos de apoyo.** Se espera que las características en cada conjunto de apoyo sean seleccionadas de manera clásica, es decir, deben diferenciar a la clase a la que pertenecen cada uno de los objetos en MA.
   Ejemplo:
   - Todos los subconjuntos de $R$ con cardinal fijo $k$, donde $k$ es un parámetro definido por el usuario.
2. **Función de semejanza parcial.** Esta función define la forma en que son comparadas los vectores de descripciones parciales de los objetos. Las descripciones parciales de los objetos, están dadas en términos de las características en los conjuntos de apoyo.
   Ejemplo:
   - $\beta(X_i^k, X_j^k) = |\{x_p \in \Omega_k | x_p^i = x_p^j\}|$. El número de características en que los objetos $O_i$ y $O_j$ coinciden.
3. **Función de evaluación parcial por fila para un conjunto de apoyo fijo.** Esta función define la manera en que cada objeto de la $MA$ vota por el objeto que se desea clasificar $O_{new}$ en términos del conjunto de apoyo $\Omega_k$. Es decir, se debe fijar la función que determine qué tan parecido o análogo es $O_{new}$ a cada uno los objetos en $MA$ tomando en cuenta su descripción parcial $X_i^{\Omega_k}$.
   Ejemplo:

- $\Gamma_k(X_i^k, X_{new}^k) = \rho(\Omega_k)\beta(X_i^k, X_{new}^k)$, donde $\rho(\Omega_k)$ es el peso asociado al conjunto de apoyo.

4. **Función de evaluación parcial por clase para un conjunto de apoyo fijo.** Esta función define la manera en que se totalizan por clases los votos parciales por filas respecto a $\Omega_k$. El objetivo de esta función es determinar qué tan parecido es $O_{new}$ a los objetos de una clase específica, considerando únicamente las características en $\Omega_k$.
   Ejemplo:

   - $\Gamma_k^j(O_{new}) = \frac{1}{n_j} \sum_{t=1}^{n_j} \Gamma_k(X_t^k, X_{new}^k)$, donde $t$ varía en los objetos de la clase $c_j$ y $n_j$ es la cantidad de objetos de la misma en $MA$.

5. **Función de evaluación total por clase para todo el sistema de conjuntos de apoyo.** Esta función define la manera en que se totaliza, para cada clase, las evaluaciones parciales en términos del sistema de conjuntos de apoyo. Mediante esta función se determina el valor de pertenencia de $O_{new}$ a una clase $c_j$.

   Ejemplo:

   - $\Phi_j(O_{new}) = \frac{1}{|\Omega^A|} \sum_{\Omega_k \in \Omega^A} \Gamma_k^j(O_{new})$, donde $\Omega^A$ es el sistema de conjuntos de apoyo seleccionado para el algoritmo $A$.

6. **Regla de solución.** Esta regla permite determinar a qué clase pertenece el objeto $O_{new}$, a partir de los valores de pertenencia de $O_{new}$ a cada clase $c_j$.
   Ejemplo:

   - Asignar a $O_{new}$ la clase $c_i$ tal que $\Phi_i(O_{new}) > \Phi_j(O_{new})$ para todo $j = 1, ..., m$ y $j \neq i$, esto es, se asigna a $O_{new}$ la clase con mayor parecido.

Se puede observar que para construir un algoritmo $A$ bajo el modelo AlVot clásico es muy versatil en el sentido de que cada etapa puede ajustarse a un problema de clasificación específico, es decir, si se tiene un amplio conocimiento del comportamiento de los objetos en $MA$, se espera que las etapas de $A$ sean lo más cercano a la realidad del problema a resolver.

Para mostrar la versatibilidad de AlVot clásico de ajustarse a un problema específico, se puede considerar a manera de ejemplo el problema de

determinar el sistema de conjuntos de apoyo de un algoritmo $A$ que permita diagnosticar si un paciente tiene un resfriado común o pulmonía. En $MA$ está la descripción de 5 pacientes con resfriado común y 4 con pulmonía. Se puede definir a cada conjunto de apoyo de acuerdo a la opinión de los médicos que trataron a los pacientes en $MA$, esto es, un médico puede determinar que para diferenciar pacientes con resfriado común de los de pulmonía, se debe conocer su temperatura corporal y la edad, otro médico puede opinar que es suficiente con la temperatura corporal.

Dado que no siempre se tiene un amplio conocimiento de los objetos en $MA$, los parámetros de $A$ también pueden ser definidos mediante técnicas automáticas. Por ejemplo, en el caso del sistema de conjuntos de apoyo, se puede utilizar un algoritmo que encuentre los testores clásicos asociados a $MA$, de esta forma cada testor correspondería a un sistema de conjuntos de apoyo.

De la descripción de las etapas, se puede observar que para clasificar un objeto nuevo $O_{new}$, el algoritmo $A$ asigna un valor numérico al voto de cada $\Omega_k$ de que $O_{new}$ pertenezca a cada una de las clases (etapa 5), se totalizan los votos de los de cada testor para una clase específica (etapa 6) y por último se asigna la clase. En la Figura 2.1 se ilustra este proceso para un problema de 3 clases, donde el sistema de conjuntos de apoyo consta de 4 elementos, cada uno de ellos vota por la simiaridad de $O_{new}$ a cada una de las 3 clases. Los votos de los 4 conjuntos de apoyo son totalizados para obtener el voto final a una clase específica, por ejemplo, para obtener el parecido final $\Phi_i(O_{new})$ a la clase $c_1$ se utizaron los valores $\Gamma_1^1(O_{new})$,$\Gamma_2^1(O_{new})$,$\Gamma_3^1(O_{new})$ y $\Gamma_4^1(O_{new})$, análogamente para obtener los parecidos finales a las clases $c_2$ y $c_3$. Por último se le asigna a $O_{new}$ la clase con la que mayor parecido tiene, es decir, la clase asociada al $\Phi_i(O_{new})$ con mayor valor.

## 2.2. Testor Clásico

El concepto de testor clásico, aunque surge relacionado con el problema particular de la detección de desperfectos en un esquema eléctrico que realiza una función booleana [Chegis and Yablonsky, 1958], ya desde la década de los sesenta [Ruiz-Shulcloperand and Abidi, 2002] se vincula a los problemas de clasificación supervisada y particularmente a los problemas de selección de características para la clasificación. En su planteamiento más simple, un testor es una combinación de características que tienen la propiedad de no confundir a objetos que pertenecen a clases diferentes.

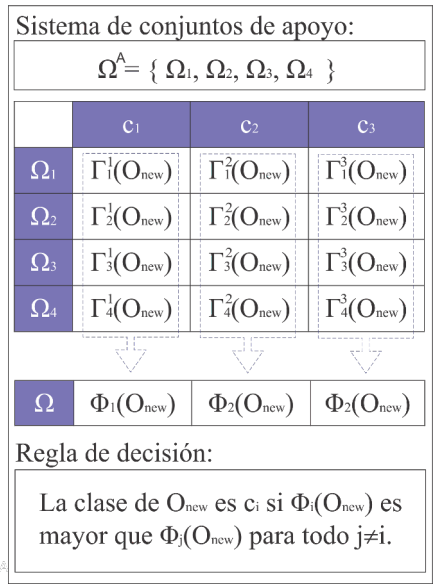


**Figura 2.1:** Ejemplo para clasificar $O_{new}$ mediante un algoritmo A basado en AlVot Clásico para un problema de 3 clases y 4 conjuntos de apoyo.

**Definición 2.1.** El conjunto de características $T = \{x_i, ..., x_s\}$ , $T \subseteq R$, se denomina *testor clásico* respecto a $MA$ si al suprimir las características de $R$ que no estén en $T$, no existe un par de filas iguales en clases diferentes.

**Definición 2.2.** El conjunto de características $T = \{x_i, ..., x_s\}$ , $T \subseteq R$, se denomina *testor típico clásico* respecto a $MA$ si es testor y no existe $T' \subset T$ tal que $T'$ sea testor.

En una $MA$ donde los objetos están descritos por $n$ características, pueden existir $2^n$ subconjuntos de características, por lo que resultaría ineficiente verificar si cada uno de ellos cumple con la definición de testor clásico (ver Definición 2.1). En la teoría de testores, para encontrar aquellos de $MA$ generalmente se trabaja con la llamada *matriz de básica* ($MB$) booleana, la cual contiene el resultado de comparaciones entre pares de objetos en $MA$ de distintas clases. La comparación se realiza característica por característica mediante un *criterio de comparación*.

**Definición 2.3.** Sea $M_i$ el conjunto de posibles valores para la característica $x_i$, se le llama *criterio de comparación*, a la función $\phi_i : M_i \times M_i \longrightarrow V$.

El criterio de comparación $\phi_i$ permite determinar qué tan *similares* o *diferentes* son dos objetos respecto a la característica $x_i$. Si se considera que

la comparación entre dos objetos se efectúa a partir de criterios de comparación definidos para cada una de las características, entonces a cada par de descripciones $(X_s, X_t)$ de objetos de $MA$ es posible asociarle un vector de comparación, aplicando dichos criterios de comparación $\phi_i$ (i=1,...,n) a cada uno de los pares de valores que toma cada característica para dichos objetos, de la siguiente forma:

$$\prod_{i=1}^{n} M_i \times \prod_{i=1}^{n} M_i \longrightarrow \prod_{i=1}^{n} V_i \tag{2.1}$$

$$(X_s, X_t) \longrightarrow (\phi_1(x_1^s, x_1^t), ..., \phi_n(x_n^s, x_n^t)) \tag{2.2}$$

Aplicando este procedimiento para todos los pares posibles de objetos que estén en clases diferentes en $MA$, se construye una *matriz de comparación* $MC$ y que en dependencia del tipo de criterios de comparación que se utilicen para cada característica, puede ser una matriz de semejanzas o de diferencias. Es decir, si los criterios de comparación definidos para cada característica permiten determinar que tan diferentes son dos objetos de acuerdo a una determinada característica, entonces se obtiene una *matriz de diferencias* $MD$. En lo siguiente, las definiciones dadas se basan en criterios de comparación de diferencia.

Cada una de las filas de $MD$ son el resultado de comparar dos objetos de $MA$ que no pertenecen a la misma clase. Si $m_i = \mid C_i' \mid$, entonces el número de filas de $MD$ está dado por:

$$m' = \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} m_i m_j \tag{2.3}$$

En general ocurre que la $MD$ contiene información redundante en el sentido de las comparaciones. Se puede apreciar que el tamaño de la matriz representa un crecimiento de orden cuadrático con respecto a $MA$, por lo que a partir de $MD$ se crea una nueva matriz en la que se elimina la información redundante (renglones que son superconjuntos de los demás). Esta reducción consiste en tomar únicamente las *filas básicas* de MD y a partir de ellas obtener una *matriz básica* $(MB)$. [Lazo-Cortes et al., 2001].

**Definición 2.4.** Sea $F_p = (a_1^p, ..., a_n^p)$ y $F_t = (a_1^t, ..., a_n^t)$ filas de una $MD$ booleana, se dice que $F_p$ es una *subfila* de $F_t$ o que $F_t$ es *superfila* de $F_p$ si, y sólo si:

- Para todo $j$ tal que $a_j^t = 0$ entonces $a_j^p = 0$

- Existe un $j_0$ tal que $a^t_{j_0} = 1$ entonces $a^p_{j_0} = 0$

**Definición 2.5.** Sea $F_t$ fila de una $MD$, se le denomina *fila básica* si y solo si en $MD$ no existe fila $F_p$ subfila de $F_t$.

**Definición 2.6.** La matriz formada por las superfilas y filas básicas de una $MD$, se le conoce como *matriz básica* respecto a $MA$ y se denota por $MB$.

Las matrices $MD$ y $MB$ tiene tantas columnas como características en $R$, sin embargo, a diferencia de $MA$, cada fila no representa las características de un objeto, sino la comparación entre dos de ellos. Cada elemento $a^j_i$ de la fila $F_j = (a^j_1, a^j_2, ..., a^j_n)$ de $MD$ o $MB$, representa el valor de la *j-ésima* comparación entre dos objetos respecto a la característica $x_i$.

En la teoría de testores, se ha demostrado que es equivalente trabajar con cualquiera de las dos matrices; $MC$ o $MB$ [Lazo-Cortes et al., 2001], debido a que el tamaño de $MB$ es menor, generalmente es la más empleada.

**Definición 2.7.** El conjunto $T = \{X_i, ..., X_s\}$ de características, $T \subseteq R$, es un *testor* respecto a $MA$ si al suprimir de $MC$ o $MB$ las columnas de $R$ que no están en $T$, no existe fila alguna $F_p$ tal que todos sus elementos son igual a 0.

---

**Ejemplo 2.1.** Sea $MA$:

| $MA$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $y$ |
|---|---|---|---|---|---|
| $O_1$ | 0 | 0 | 1 | 0 | $c_1$ |
| $O_2$ | 1 | 1 | 0 | 0 | $c_1$ |
| $O_3$ | 0 | 0 | 0 | 1 | $c_2$ |
| $O_4$ | 1 | 1 | 1 | 1 | $c_2$ |
| $O_5$ | 1 | 0 | 1 | 1 | $c_3$ |

Utilizando para cada característica como criterio de comparación la desigualdad (1, si $x^s_i \neq x^t_i$ y 0 si son iguales), se obtiene la siguiente $MD$:

| $MD_1$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ |
|---|---|---|---|---|
| $O_1O_3$ | 0 | 0 | 1 | 1 |
| $O_1O_4$ | 1 | 1 | 0 | 1 |
| $O_1O_5$ | 1 | 0 | 0 | 1 |
| $O_2O_3$ | 1 | 1 | 0 | 1 |
| $O_2O_4$ | 0 | 0 | 1 | 1 |
| $O_2O_5$ | 0 | 1 | 1 | 1 |

Como puede apreciarse en este caso $m' = 2(1+2)+1(2) = 8$.

La $MB$ resultante es:

| $x_1$ | $x_2$ | $x_3$ | $x_4$ |
|---|---|---|---|
| 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |

Y para ella se tienen los siguientes testores:

$\{x_1, x_2, x_3, x_4\}$

| | | | |
|---|---|---|---|
| $\{x_1, x_2, x_3\}$ | $\{x_1, x_3, x_4\}$ | $\{x_1, x_2, x_4\}$ | $\{x_1, x_2, x_3\}$ |
| $\{x_3, x_4\}^*$ | $\{x_1, x_4\}^*$ | $\{x_1, x_3\}^*$ | |

Los marcados con * son típicos.

---

## 2.3. Testor por clase

Como se ha mencionado, el concepto de testor está vinculado a problemas de selección de características, donde el objetivo es encontrar las características de los objetos que permiten diferenciar a los objetos de distintas clases. Este problema puede ser abordado de dos posibles maneras: considerando en la selección todas las clases o considerando una clase específica [Pineda-Bautista et al., 2011]; surgiendo de esta última los *testores por clase* [J. Martínez-Trinidad and Contreras-Arévalo, 2000].

**Definición 2.8.** Sea $T \subseteq R$, $T$ se considera *testor por clase* para la clase $c_i$ si al suprimir de $MA$ las características de $R$ que no estén en $T$, no existen filas con clase $c_i$, iguales a cualquiera de las restantes clases.

**Definición 2.9.** Sea $T \subseteq R$, $T$ se considera *testor típico por clase* para la clase $c_i$ si es testor por clase de la clase $c_i$ y no existe $T' \subset T$ tal que $T'$ sea testor por clase para la clase $c_i$.

Los testores por clase son un conjunto de características que permiten diferenciar objetos de una clase específica con los del resto de las clases. La matriz de diferencias de la clase $c_i$ se denota como $MD_i$, para $i = 0, ..., m$, donde las filas de cada $MD_i$ corresponden a las comparaciones que involucran

a los objetos de $c_i$ con los de las clases restantes. Se tiene ahora que la cantidad de filas para la $MD_i$ está dada por:

$$m'_i{=}m_i \sum_{j \neq i} m_j \tag{2.4}$$

Cada matriz $MD_i$ cumple con la propiedad de ser subconjunto de la $MD$ en el sentido clásico, es decir $MD_i \subseteq MD$, para todo $i = 0, ..., m$. Las definiciones de subfila, superfila y fila básica (ver Definiciones 2.4 y 2.5) son válidas para cada $MD_i$, por lo tanto para cada una de ellas se tienen sus respectivas matrices básicas $MB_1, ..., MB_m$.

---

**Ejemplo 2.2.** Utilizando la misma $MA$ del Ejemplo 2.1 se tienen las siguientes matrices de diferencias:

| $MD_1$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ |
|---|---|---|---|---|
| $O_1O_3$ | 0 | 0 | 1 | 1 |
| $O_1O_4$ | 1 | 1 | 0 | 1 |
| $O_1O_5$ | 1 | 0 | 0 | 1 |
| $O_2O_3$ | 1 | 1 | 0 | 1 |
| $O_2O_4$ | 0 | 0 | 1 | 1 |
| $O_2O_5$ | 0 | 1 | 1 | 1 |

| $MD_2$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ |
|---|---|---|---|---|
| $O_3O_1$ | 0 | 0 | 1 | 1 |
| $O_3O_2$ | 1 | 1 | 0 | 1 |
| $O_3O_4$ | 1 | 1 | 1 | 0 |
| $O_3O_5$ | 1 | 0 | 1 | 0 |

| $MD_3$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ |
|---|---|---|---|---|
| $O_4O_1$ | 1 | 1 | 0 | 1 |
| $O_4O_2$ | 0 | 0 | 1 | 1 |
| $O_4O_3$ | 1 | 1 | 1 | 0 |
| $O_5O_1$ | 1 | 0 | 0 | 1 |
| $O_5O_2$ | 0 | 1 | 1 | 1 |
| $O_5O_3$ | 1 | 0 | 1 | 0 |

Para cada $MD_i$ se tienen las siguientes matrices básicas:

| $MB_1$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ |
|---|---|---|---|---|
| | 0 | 0 | 1 | 1 |
| | 1 | 1 | 0 | 1 |

| $MB_2$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ |
|---|---|---|---|---|
| | 0 | 0 | 1 | 1 |
| | 1 | 1 | 0 | 1 |
| | 1 | 1 | 1 | 1 |

| $MB_3$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ |
|---|---|---|---|---|
| | 1 | 1 | 0 | 1 |
| | 0 | 0 | 1 | 1 |
| | 1 | 1 | 1 | 0 |
| | 1 | 0 | 0 | 1 |
| | 0 | 1 | 1 | 1 |
| | 1 | 0 | 1 | 0 |

Los testores por clase para la clase $c_1$ son:

$\{x_1, x_2, x_3, x_4\}$
$\{x_2, x_3, x_4\}$ $\{x_1, x_3, x_4\}$ $\{x_1, x_2, x_4\}$ $\{x_1, x_2, x_3\}$
$\{x_1, x_4\}$ $\{x_3, x_4\}$ $\{x_2, x_4\}$ $\{x_1, x_3\}^*$
$\{x_4\}^*$

Los testores por clase para la clase $c_2$ son:

$\{x_1, x_2, x_3, x_4\}$
$\{x_2, x_3, x_4\}$ $\{x_1, x_3, x_4\}$ $\{x_1, x_2, x_4\}$ $\{x_1, x_2, x_3\}$
$\{x_1, x_4\}^*$ $\{x_3, x_4\}^*$ $\{x_1, x_3\}^*$ $\{x_2, x_3\}^*$

Los testores por clase para la clase $c_3$ son:

$\{x_1, x_2, x_3, x_4\}$
$\{x_2, x_3, x_4\}$ $\{x_1, x_3, x_4\}$ $\{x_1, x_2, x_4\}$ $\{x_1, x_2, x_3\}$
$\{x_1, x_4\}^*$ $\{x_3, x_4\}^*$ $\{x_1, x_2\}^*$

Los testores por clase marcados con * son típicos.

---

Con base en los Ejemplos 2.1 y 2.2 se puede observar que:

- En la $MD$ no hay dos filas iguales en clases diferentes por lo que se tiene que $\{x_1, x_2, x_3, x_4\}$ es testor clásico y también es testor por clase para cada una de las clases, sin embargo $\{x_4\}$ es testor típico por clase para la clase $C_1$, sin embargo no es testor clásico ni testor por clase para el resto de las clases.
- Los testores clásicos también son testores por clase para cada clase.
- Todos los testores típicos por clase que a la vez son testores clásicos para $MA$, son testores típicos clásicos.

## 2.4. AdaBoost

AdaBoost [Freund and Schapire, 1995] es un algoritmo para realizar un ensamble de clasificadores. La idea básica de un ensamble es crear una única hipótesis de clasificación altamente precisa al combinar varias hipótesis de clasificación con baja precisión -obtenidas de un *clasificador débil*-. Esto es, combinar clasificadores con bajo porcentaje de objetos clasificados correctamente, para construir uno con un porcentaje de clasificación alto.

Existen dos técnincas para realizar un ensamble de clasificadores *bagging* y *boosting*. En la primera los objetos en la muestra de aprendizaje son separados aleatoriamente (con o sin remplaso) para formar $K$ matrices de aprendizaje, cada matriz formada será asignada como $MA$ para un clasificador debil y despues sus hipótesis son combinadas para clasificar correctamente la mayor cantidad de objetos posible. A diferencia del ensamble mediante *bagging*, en el *boosting* se utilizan todos los objetos de $MA$ en cada uno de los clasificadores débiles. AdaBoost [Freund and Schapire, 1995] fue el primer algoritmo práctico de *boosting*.

Continuando con la notación descrita a lo largo de este capítulo, se muestra en el Algoritmo 2.1 el pseudocódigo de AdaBoost para un problema de clasificación de dos clases que para fines prácticos se denotan como *clase positiva* y *clase negativa*, donde:

- $D_k(i)$ denota el peso asignado al objeto $O_i$ en $MA$ en la iteración $k$, los cuales cumplen con la siguiente propiedad:

$$\sum_{i=1}^{m} D_k(i) = 1 \tag{2.5}$$

- $f$ se describe como:

$$f(y_i) = \begin{cases} +1 & si \quad y_i = 1 \\ -1 & si \quad y_i = 0 \end{cases} \tag{2.6}$$

- $Z_k$ es un factor de normalización elegido de manera que $D_{k+1}$ cumpla con la propiedad mostrada en la Ecuación 2.5

La salida de AdaBoost es una hipótesis $H$ que determina si un objeto $O$ con vector de características $X$, pertenece a la clase positiva o a la negativa, esto es, el valor de $H(X)$ es $+1$, si $X$ pertenece a la clase positiva, y es $-1$ si pertenece a la clase negativa.

**Algoritmo 2.1.** Pseudocódigo de AdaBoost.

---

**Entrada:** $MA$ ($n$ es el número de objetos en $MA$)
**para** $i = 1, ..., n$ **hacer**
  $D_1(1) = \frac{1}{n}$
**fin para**
**para** $k = 1, ..., K$ **hacer**
  Usando la distribución $D_k$, entrenar a los clasificadores débiles.
  Obtener las hipótesis débiles $h_k$.
  **para todo** $h_k$ **hacer**
    **para** $i = 1, ..., n$ **hacer**
      $\varepsilon = \Pr_{i \sim D_k}[h_k(X_i) \neq f(y_i)]$
    **fin para**
  **fin para**
  Seleccionar $h_k$ tal que su error $\varepsilon$ sea menor.
  $\alpha_k = \frac{1}{2} \ln(\frac{1-\varepsilon_k}{\varepsilon_k})$.
  **para** $i = 1, ..., n$: **hacer**
    $D_{k+1}(i) = \frac{D_k(i) \exp(-\alpha_k f(y_i) h_k(X_i))}{Z_k}$
  **fin para**
  Construir hipótesis fuerte:
  $H(X_i) = \text{sign}(\sum_{k=1}^{K} \alpha_k h_k(X))$
**fin para**

---

El algoritmo AdaBoost comienza seleccionando al clasificador débil con más objetos clasificados correctamente, en la siguiente iteración selecciona a aquel clasificador que mejor clasifique a los objetos que el anterior hizo incorrectamente. Continuando de esta manera se construye un clasificador fuerte. Es importante notar que AdaBoost asigna en cada iteración un peso mayor a los objetos que no han sido correctamente clasificados en la iteración anterior, esto con el propósito de que los errores sean mayores para los clasificadores débiles que los clasifiquen de manera incorrecta, lo cual se ilustra con el Ejemplo 2.3.

---

**Ejemplo 2.3.** Considerar una matriz de aprendizaje que contiene 5 puntos de clase positiva y 5 de clase negativa. En la figura 2.2 se muestran cuatro clasificadores débiles que tienen como hipotesis que los puntos en el área azul son de clase positiva y el resto de clase negativa.

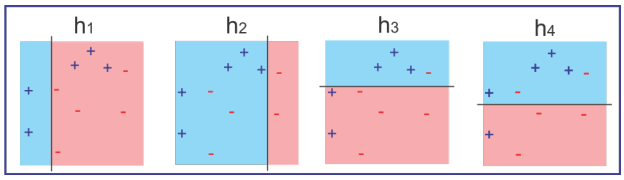


**Figura 2.2:** Hipótesis de 4 clasificificadores débiles.

En cada iteración AdaBoost selecciona al mejor clasificador, tal como se muestra en Figura 2.3. En la iteración k=1 tanto $h_1$,$h_2$ y $h_3$ son los que cometen menos errores, pero selecciona al primero, es decir, a $h_1$. En la siguiente iteración k=2 debe seleccionar a aquel que mejor clasifique, considerando que los puntos que $h_1$ no clasificó correctamente tienen mayor peso, de este modo AdaBoost selecciona a $h_2$. En la última iteración sólo puede elegir entre $h_3$ y $h_4$, dado que $h_3$ tiene menor error, es seleccionado. En la Figura 2.4 se muestra el clasificador fuerte construido por AdaBoost, donde se puede apreciar que su hipótesis no comete ningún error.

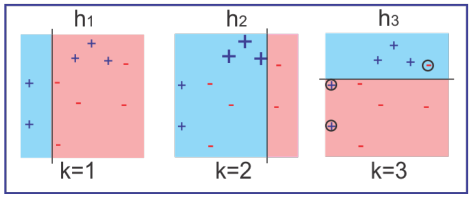


**Figura 2.3:** Hipótesis de los clasificadores de la Figura 2.2 seleccionadas por AdaBoost en la iteración $k = 1$, $k = 2$ y $k = 3$.

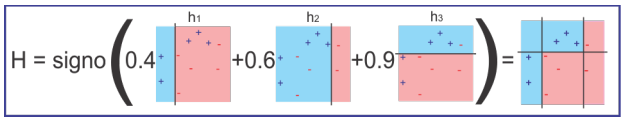


**Figura 2.4:** Hipótesis del clasificador fuerte contruido por AdaBoost a partir de los clasificadores débiles de la Figura 2.2.

## 2.5. Trabajo relacionado

En problemas de clasificación supervisada, desde finales de la década de los 90's se presenta la idea de trabajar con las características de los objetos

por clase específica. En [Baggenstoss, 1998, Baggenstoss, 1999] se plantea que la tarea de estimar las funciones de densidad de probabilidad (fdp) para un clasificador bayesiano se facilita al considerar cada fdp para valores específicos de cada una de las clases en vez de para todas.

La selección de características por clase está fuertemente ligada al uso de un clasificador particular, por ejemplo, en [A. et al., 2013] proponen utilizar una red hiperesferas. Cada una de ellas tiene como centro a un objeto de $MA$ -seleccionado aleatoriamente- con radio igual a la distancia entre el objeto centro y el más lejano que cumpla con la condición de que la mayoría de objetos dentro de ella sean de la misma clase que la del centro. Además, una esfera tiene asociada un subconjunto de características seleccionadas por clase -de acuerdo a la clase de su centro-, elegidas de acuerdo a su *índice de separabilidad*. Se selecciona el subconjunto de características con mayor índice de separabilidad, este se calcula dividiendo la media de las distancias interclase entre las intraclase de los objetos dentro de la esfera. La clase de un nuevo objeto se determina de dos posibles maneras, la primera es asignarle la clase de la hiperesfera que lo contenga y la segunda sólo ocurre cuando ninguna hiperesfera lo contiene, asignándole la misma clase del objeto más cercano.

En [Pineda-Bautista et al., 2011] se propone un esquema de selección de características por clase en el que se puede utilizar cualquier clasificador. El esquema consiste en cuatro etapas: binarización de clases, balanceo de clases, selección de características por clase y clasificación (ver Figura 2.5). Esta última etapa no forma parte de la selección de características, sin embargo, los autores la definen para mostrar la ventaja de la selección de características por clase utilizando clasificadores convencionales que sólo utilizan un conjunto de características. A continuación se describen cada una de las etapas:

**Binarización de clases.** En esta etapa se utiliza la binarización *una clase contra todas* propuesta en [ref], donde el problema de clasificación de $m$ clases es dividido en $m$ problemas de dos clases. Esto es, para cada clase $c_i$,con $i = 1, \ldots, m$ un problema binario $\langle c_i, c_i^{'} \rangle$ es creado, donde $c_i^{'} = \bigcup_{j=1}^{m} c_j$ y $j \neq i$ (ver Figura 2.6).

**Balanceo de clases.** Al utilizar la binarización *una clase contra todas*, los problemas binarios que se generan podrían estar desbalanceados. Esto es, en el problema binario $\langle c_i, c_i^{'} \rangle$, el número de objetos $c_i$ puede resultar menor que

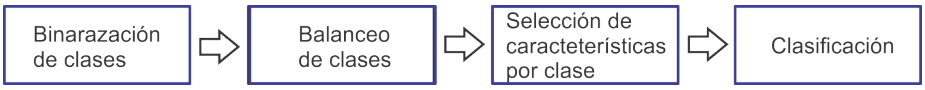


**Figura 2.5:** Etapas en el esquema de selección de características por clase propuesto en [Pineda-Bautista et al., 2011]

el de los de la clase $\Theta_i$. Para solucionar este problema los autores proponen realizar sobremuestreo repitiendo objetos de $c_i$ hasta alcanzar el balance, es decir, $\mid c_i \mid = \mid c_i^{'} \mid$.

**Selección de características por clase.** Esta etapa consiste en seleccionar las características de cada problema binario, esto mediante la utilización de selectores de características tradicionales. Al realizar esta tarea, las características seleccionadas para el problema $\langle c_i, c_i^{'} \rangle$ corresponden a las características para la clase $c_i$.

**Clasificación.** Una vez seleccionadas las características para cada una de las clases, se debe determinar un clasificador $e_i$, el cual tendrá como muestra de aprendizaje a aquella que corresponde al problema original de $m$ clases, pero considerando para cada objeto, únicamente las características asociadas a la *i-ésima* clase. Dado un nuevo objeto $O_{new}$, cada clasificador $e_i$ le asigna una clase a dicho objeto, para finalmente determinar a qué clase pertenece $O_{new}$, los autores proponen una *regla de decisión*. Esta considera dos casos: el *i-ésimo* clasificador $e_i$ le asigna a $O_{new}$ la clase $c_i$, para cualquier $i \leq m$ ;no existe clasificador $e_i$ tal que le asigne a $O_{new}$ la clase $c_i$. En el primer caso, se le asigna a $O_new$ la clase $c_i$, en caso de empate se le asigna la mayoría de votos, si el empate continúa entonces se le asigna aquella clase -involucrada en el empate- con mayor número de objetos. El segundo caso, asigna a $O_{new}$ la clase con mayoría de votos, si este continúa le asigna la clase -también involucrada en el empate- con mayor número de objetos. En la Figura 2.7 se muestra el proceso a seguir en esta etapa.

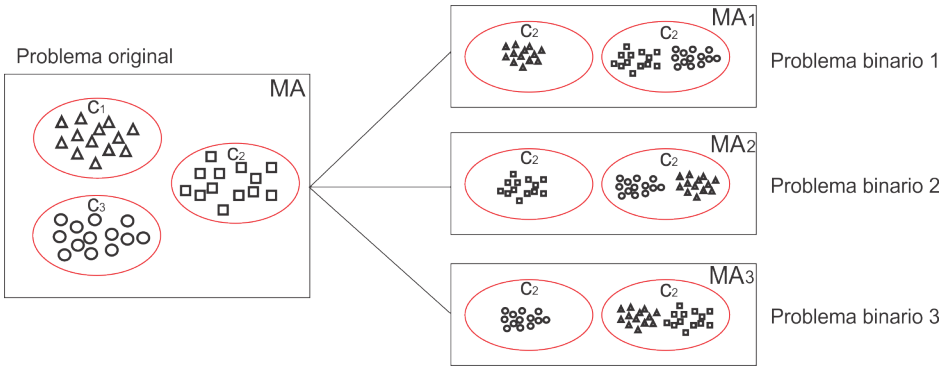


**Figura 2.6:** Ejemplo de *una clase contra todas* para un problema de clasificación supervisada de tres clases.

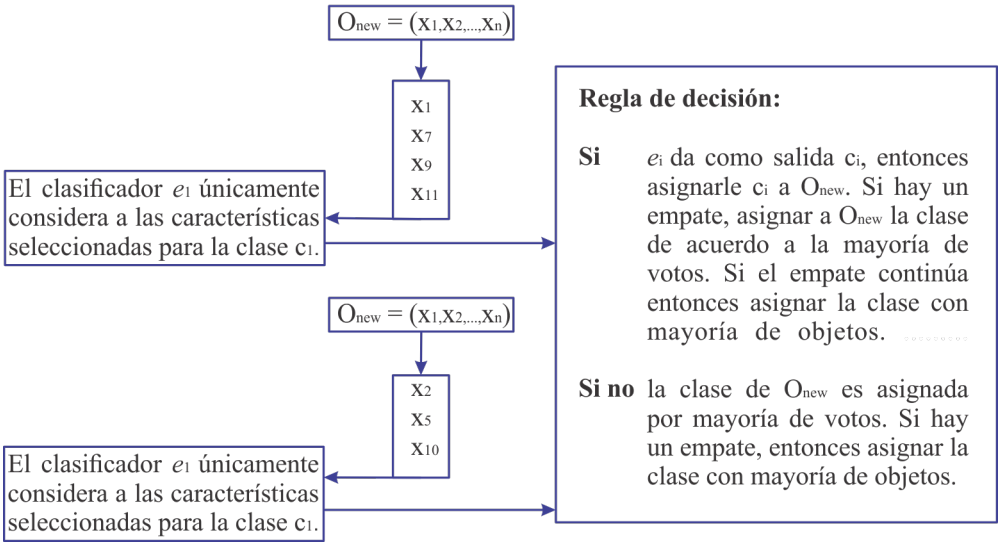


**Figura 2.7:** Proceso de clasificación propuesto en [Pineda-Bautista et al., 2011] utilizando selección de características por clase.

# Capítulo 3

# AlVot con un sistema de conjuntos de apoyo por clase

En este capítulo se define AlVot PC, incluyendo un apartado para definir un algoritmo basado en AlVot PC el cual utiliza a los testores por clase como sistemas de conjuntos de apoyo. En otro apartado se explica la manera de utilizar AdaBoost para seleccionar a los testores por clase. Por último, se incluye una sección donde se define la manera en que se obtienen los testores por clase a partir de los criterios de comparación propuestos.

## 3.1. AlVot PC

En la Sección 2.1 se describió AlVot clásico, el cual requiere de fijar 6 parámetros para definir un algoritmo. A diferencia de AlVot Clásico, AlVot PC considera un sistema de conjuntos de apoyo para cada una de las clases, por esta razón es necesario realizar ajustes a Alvot clásico. Para definir las estapas de AlVot PC se debe tomar en cuenta la clase complemento de cada una de las clases originales, es decir, se binarizan las clases mediante el método *una clase contra todas* descrito en la Sección 2.5 (ver Figura 2.6).

A diferencia de AlVot clásico, AlVot PC considera 7 que para definirlas se utiliza la notación:

- $A$. Algoritmo basado en AlVot PC.
- $\Omega_i^A$. Sistema de conjuntos de apoyo de $A$ de la i-ésima clase.
- $n$. Número de características que describen a los objetos en $MA$.

- $R = \{x_1, x_2, ..., x_n\}$. Conjunto de características que describe a los objetos en $MA$.
- $\Omega_k^i = \{x_p, ..., x_s\}$. k-ésimo conjunto de apoyo de la i-ésima clase, donde $1 \leq p \leq n$, $1 \leq s \leq n$, $\Omega_k^i \subseteq R$ y $\Omega_k^i \in \Omega_i^A$.
- $X_j$. *Vector de descripción total* del objeto $O_j$, de acuerdo a las características en $R$.
- $X_j^{k_i}$. *Vector de descripción parcial* del objeto $O_j$, de acuerdo a las características en el conjunto de apoyo $\Omega_k^i$.
- $x_p^j$. Valor de la característica $x_p \in R$ del objeto $O_j$.
- $m$. Número de clases en $MA$.
- $C = \{c_1, ..., c_m\}$. Conjunto de valores para las clases.
- $C' = \{c_1^{'}, ..., c_m^{'}\}$. Conjunto de valores para las clases generadas en la binarización de $MA$.
- $y_j$. Valor de la clase del objeto $O_j$, $y_j \in C$.

Las 7 etapas se describen de la siguiente manera:

1. **Sistema de conjuntos de apoyo para una clase específica.** Se espera que las características en cada conjunto de apoyo sean seleccionadas de manera que diferencien a la objetos de clase $c_j$ y los de $c_j^{'}$.
   Ejemplo:
   - Todos los subconjuntos de $R$ con cardinal fijo $k$, donde $k$ es un parámetro definido por el usuario.
2. **Función de semejanza parcial.** Esta función está definida de la misma forma que en AlVot Clásico.
   Ejemplo:
   - $\beta(X_i^{k_q}, X_j^{k_q}) = |\{x_p \in \Omega_k^q | x_p^i = x_p^j\}|$. El número de características en que los objetos $O_i$ y $O_j$ coinciden.
3. **Función de evaluación parcial por fila para un conjunto de apoyo fijo.** Esta función está definida de la misma forma que en AlVot Clásico.
   Ejemplo:

- $\Gamma_{k_q}(X_i^{k_q}, X_{new}^{k_q}) = \rho(\Omega_k^q)\beta(X_i^{k_q}, X_{new}^{k_q})$, donde $\rho(\Omega_k^q)$ es el peso asociado al conjunto de apoyo.

4. **Función de evaluación parcial por clase y su complemento para un conjunto de apoyo fijo.** El objetivo de esta función es determinar qué tan parecido es $O_{new}$ a los objetos de una clase específica $c_j$ y a los de su complementno $c_j^{'}$.
Ejemplo:

   - $\Gamma_k^j(O_{new}) = \frac{1}{n_j}\sum_{i=1}^{n_j}\Gamma_{k_j}(X_i^{k_j}, X_{new}^{k_j})$, donde $i$ varía en los objetos de la clase $c_j$ y $n_j$ es la cantidad de objetos de la misma.
   - $\Gamma_k^{'j}(O_{new}) = \frac{1}{n_j^{'}}\sum_{i=1}^{n_j}\Gamma_{k_j}(X_i^{k_j}, X_{new}^{k_j})$, donde $i$ varía en los objetos de la clase $c_j^{'}$ y $n_j^{'}$ es la cantidad de objetos de la misma.

5. **Función de evaluación total por clase y su complemplemento para un sistema de conjuntos de apoyo específico.** Esta función define la manera en que se totaliza, para una clase y su complemento, las evaluaciones parciales en términos del sistema de conjuntos de apoyo $\Omega_j^A$.
Ejemplo:

   - $\Phi_j(O_{new}) = \frac{1}{|\Omega_j^A|}\sum_{\Omega_k^j \in \Omega_j^A}\Gamma_k^j(O_{new})$.
   - $\Phi_j^{'}(O_{new}) = \frac{1}{|\Omega_j^A|}\sum_{\Omega_k^j \in \Omega_j^A}\Gamma_k^{'j}(O_{new})$.

6. **Función de evaluación total por clase para un sistema de conjuntos de apoyo específico.** Mediante esta función se determina el valor de pertenencia de $O_{new}$ a la clase $c_j$. La función contempla los valores $\Phi_j(O_{new})$ y $\Phi_j^{'}(O_{new})$.
Ejemplo:

   - $\lambda_j(O_{new}) = \Phi_j(O_{new}) - \Phi_j^{'}(O_{new})$

7. **Regla de solución.** Esta regla se define igual que en AlVot Clásico.
Ejemplo:

   - Asignar a $O_{new}$ la clase $c_i$ tal que $\lambda_i(O_{new}) > \lambda_j(O_{new})$ para todo $j = 1, ..., m$ y $j \neq i$, esto es, se asigna a $O_{new}$ la clase con mayor parecido.

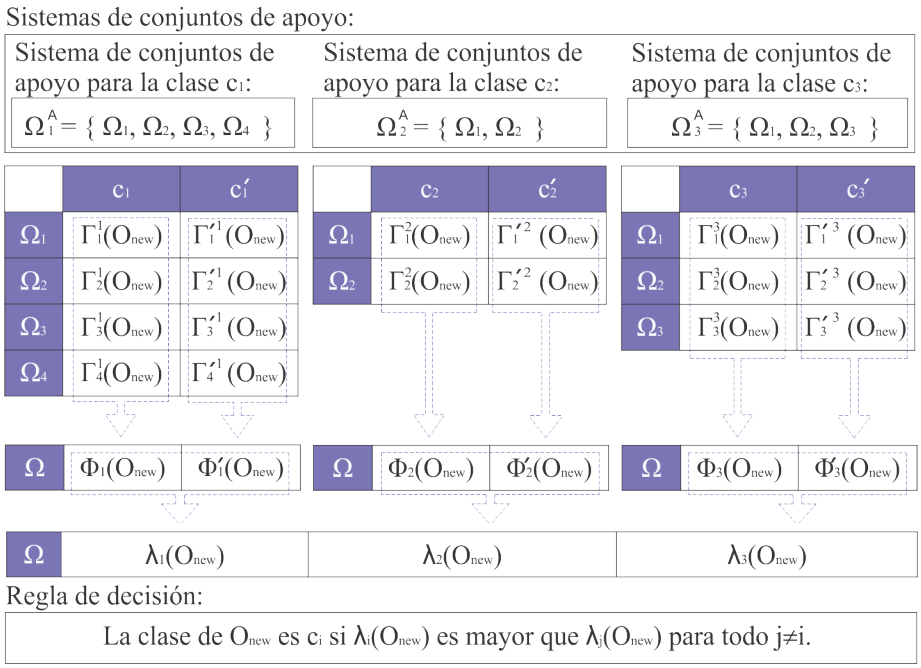


**Figura 3.1:** Ejemplo para clasificar $O_{new}$ mediante un algoritmo $A$ basado en AlVot PC para un problema de 3 clases con 4, 2 y 3 conjuntos de apoyo en las clases $c_1$,$c_2$ y $c_3$ respectivamente.

Para facilitar la comprensión del modelo propuesto, en la Figura 3.1 se presenta un ejemplo de un algoritmo basado en AlVot PC para un problema de 3 clases donde se aprecia que cada clase tiene un sistema de conjuntos apoyo ($\Omega_1^A$,$\Omega_2^A$ y $\Omega_3^A$). Para cada clase, todo conjunto de apoyo vota por el parecido a su clase asociada y con la clase complemento. Por ejemplo, $\Gamma_1^3(O_{new})$ y $\Gamma_1'^3(O_{new})$ representan el parecido de $O_{new}$ a los objetos de clase $c_3$ y su clase complemento $c_3'$ respectivamente de acuerdo al conjunto de apoyo $\Omega_1^3$. Los valores $\Phi_3(O_{new})$ y $\Phi_3'(O_{new}$ representan la totalización de los votos de $\Omega_1^3$,$\Omega_2^3$ y $\Omega_2^3$, es decir, del sistema de conjuntos de apoyo $\Omega_3^A$. En $\lambda_j(O_{new})$ se representa el parecido final de $O_{new}$ a los objetos de clase $c_3$, este valor no toma en cuenta unicamente a $\Phi_3(O_{new})$ sino también a $\Phi_3'(O_{new}$ porque por ejemplo, si $\Phi_3(O_{new})$ tiene un valor muy alto, quiere decir que $O_{new}$ es muy parecido a la clase $c_3$ y sería de espererarse que $\Phi_3'(O_{new})$ tenga un valor muy bajo, si esto ocurre aumenta la prosibilidad de que $O_{new}$ pertenezca a la clase $c_3$. Por último se decide la clase de $O_{new}$ con la clase asociada al $\Phi_i(O_{new})$ con mayor valor.

## 3.2. Algoritmo basado en AlVot PC

AlVot PC es un modelo que permite la construcción de algoritmos de clasificación supervizada. A continuación se propone cómo definir cada una de sus etapas:

1. **Sistema de conjuntos de apoyo para una clase específica.**
   - $\Omega_j^A$ =Testores típicos para la clase $c_j$.
2. **Función de semejanza parcial.**
   - $\beta(X_i^{k_q}, X_j^{k_q}) = 1 - HEOM(X_i^{k_q}, X_j^{k_q})$

   donde:
   - $HEOM(X_i^{k_q}, X_j^{k_q}) = \sqrt{\sum_{x_p \in \Omega_k^q} d(x_p^i, x_p^j)^2}$
   - $d(x_p^i, x_p^j) = \begin{cases} 1 & si \quad x_p^i \vee x_p^j \quad es\ faltante \\ overlap(x_p^i, x_p^j) & si \quad x_p \quad es\ numerica \\ rn\_diff(x_p^i, x_p^j) & si \quad x_p \quad es\ nominal \end{cases}$
   - $overlap(x_p^i, x_p^j) = \begin{cases} 0 & si \quad x_p^i = x_p^j \\ 1 & en\ otro\ caso. \end{cases}$
   - $rn\_diff(x_p^i, x_p^j) = \frac{|x_p^i - x_p^j|}{rango(x_p)}$
   - $rango(x_p) = max(x_p) - min(x_p)$

   Para construir un algoritmo A basado en AlVot PC, se deben determinar las etapas descritas a continuación:

3. **Función de evaluación parcial por fila para un conjunto de apoyo fijo.**
   - $\Gamma_{k_j}(X_i^k, X_{new}^k) = \beta(X_i^k, X_{new}^k)$
4. **Función de evaluación parcial por clase y su complemento para un conjunto de apoyo fijo.**

- $\Gamma_k^j(O_{new}) = \frac{1}{5}\sum_{i=1}^{5}\Gamma_{k_j}(X_i^{k_j}, X_{new}^{k_j})$, donde $i$ varía en los 5 objetos de la clase $c_j$ más similares a $O_new$ respecto a $\Omega_k^j$.
- $\Gamma_k^{'j}(O_{new}) = \frac{1}{5}\sum_{i=1}^{5}\Gamma_{k_j}(X_i^{k_j}, X_{new}^{k_j})$, donde $i$ varía en los 5 objetos de la clase $c_j^{'}$ más similares a $O_new$ respecto a $\Omega_k^j$.

5. **Función de evaluación total por clase y su complemplemento para un sistema de conjuntos de apoyo específico.**
   - $\Phi_j(O_{new}) = \frac{1}{|\Omega_j^A|}\sum_{\Omega_k^j \in \Omega_j^A}\Gamma_k^j(O_{new})$.
   - $\Phi_j^{'}(O_{new}) = \frac{1}{|\Omega_j^A|}\sum_{\Omega_k^j \in \Omega_j^A}\Gamma_k^{'j}(O_{new})$.
6. **Función de evaluación total por clase para un sistema de conjuntos de apoyo específico $\lambda_j(O_{new})$.**
   - $\lambda_j(O_{new}) = \Phi_j(O_{new}) + (1 - \Phi_j^{'}(O_{new}))$
7. **Regla de solución.**
   - Asignar a $O_{new}$ la clase $c_i$ tal que $\lambda_i(O_{new}) > \lambda_j(O_{new}))$ para todo $j = 1, ..., m$ y $j \neq i$, esto es, se asigna a $O_{new}$ la clase con mayor paracido.

Continuando con la notación descrita en este capítulo, al definir de esta forma las etapas de AlVot PC, los pasos a seguir para clasificar un nuevo objeto $O_{new}$ se describen en el Algoritmo 3.1.

**Algoritmo 3.1.** Pseudocódigo del algoritmo propuesto basado en AlVot PC.

---

**Entrada:** $O_{new}$,$MA$

```
para todo c_i en MA hacer
    Φ_i(O_new) = Φ'_i(O_new) = 0
    Ω_i^A = obtenerTestoresTipicos(MAbin(i))
    para todo Ω_j^i en Ω_i^A hacer
        para todo O_k en MAbin(i) hacer
            si y_k = c_i entonces
                voto(i)(j)(k) = 1 − HOEM(Onew, O_k)
```

**si no**
$voto'(i)(j)(k) = 1 - HOEM(Onew, O_k)$
**fin si**
**fin para**
$\Gamma^i_j(O_{new}) =$ *obtenerPromedioDeLos5Mayores*$(voto(i)(j))$
$\Gamma'^i_j(O_{new}) =$ *obtenerPromedioDeLos5Mayores*$(voto'(i)(j))$
$\Phi_i(O_{new}) = \Phi_i(O_{new}) + \Gamma^i_j(O_{new})$
$\Phi'_i(O_{new}) = \Phi'_i(O_{new}) + \Gamma'^i_j(O_{new})$
**fin para**
$\Phi_i(O_{new}) = \Phi_i(O_{new})/|\Omega^A_i|$
$\Phi'_i(O_{new}) = \Phi'_i(O_{new})/|\Omega^A_i|$
$\lambda_i(O_{new}) = \Phi_i(O_{new}) + (1 - \Phi'_i(O_{new}))$
**fin para**
$win =$*argmax*$(\lambda_i(O_{new}))$ para $i = 1, \ldots, m$
**retornar** $c_{win}$

---

## 3.3. Algoritmo basado en PC con AdaBoost como selector de testores por clase

En la Sección 2.4 se describió el algoritmo de AdaBoost. En el contexto de este proyeto, el propósito de utilizar AdaBoost es seleccionar y ponderar los testores por clase en cada sistema de conjuntos de apoyo del algoritmo propuesto basado en AlVot PC. Para realizar dicha tarea se deben considerar los $m$ problemas binarios generados al aplicar el método *una clase contra todas* a $MA$, además de un algoritmo $A$ basado en AlVot clásico análogo al propuesto en la Sección 3.2.

Recordar que los testores por clase de una clase $c_i$, permiten diferenciar a los objetos en dicha clase respecto a los de su clase complemento $c^i_i$. Cada clasificador débil en AdaBoost es el algoritmo $A$ que considera a un único testor por clase como sistema de conjuntos de apoyo y las clases positiva y negativa, corresponden a $c_i$ y $c^i_i$ respectivamente. A manera de ejemplo, si la clase $c_i$ tiene 6 testores, entonces se tiene la misma cantidad de instancias de $A$, denotadas como $A_1, \ldots, A_6$, cada una con un testor asociado.

Bajo este contexto y continuando con el ejemplo, al ejecutar AdaBoost, el resultado es una combinación lineal de un subconjunto de las instancias de $A$, esto es, $\alpha_j A_i + ... + \alpha_j A_j$. Si dicha combinación lineal está formada únicamente por 3 de las 6 instancias de $A$, entonces quiere decir que el

resto resultan innecesarias. Dado que cada $A_i$ en la combinación lineal tiene asociado un testor y un peso, se propone eliminar del sistema de conjunto de apoyo a los testores que AdaBoost no considera.

Es importante notar que cada instancia de $A$ seleccionada por AdaBoost tiene un peso $\alpha$ asociado, se propone utilizar dicho peso al totalizar los votos del nuevo sistema de conjuntos de apoyo, esto es para ponderar cada conjunto de apoyo. Cuado se seleccionan y ponderan los testores por clase de un sistema de conjuntos de apoyo, se define un nuevo algoritmo basado en AlVot PC que a diferencia del propuesto en la sección anterior, se modifica únicamente la etapa 1 y 5 descritas a continuación:

1. **Sistema de conjuntos de apoyo para una clase específica.**
   - $\Omega_j^A$ =Testores típicos para la clase $c_j$ seleccionados por AdaBoost.

5. **Función de evaluación total por clase y su complemplemento para un sistema de conjuntos de apoyo específico.**
   - $\Phi_j(O_{new}) = \frac{1}{\sum_{i=1}^{s} \alpha_i} \sum_{\Omega_k^j \in \Omega_j^A} \alpha_k \Gamma_k^j(O_{new})$
   - $\Phi_j^{'}(O_{new}) = \frac{1}{\sum_{i=1}^{s} \alpha_i} \sum_{\Omega_k^j \in \Omega_j^A} \alpha_k \Gamma_k^{'j}(O_{new})$

   donde $\alpha_i$ es el peso asociado al conjunto de apoyo $\Omega_i^j$ asignado por AdaBoost y $s$ es el número de conjuntos de apoyo en $\Omega_j^A$.

Se muestra en el Algoritmo 3.2 el pseudocódifo de redefinir de esta manera las etapas de AlVot PC, donde se aprecian dos diferencias principales respecto al Algoritmo 3.1; en la linea 5 seleccionan los testores al llamar a AdaBoost y que las lineas 15 y 16 se utiliza el peso asignado por AdaBoost para contabilizar el voto de cada testor, como resultado, en las lineas 18 y 19 se obtiene el promedio ponderado de los votos de cada testor.

**Algoritmo 3.2.** Pseudocódigo del algoritmo propuesto basado en AlVot PC que utiliza AdaBoost para seleccionar a los sistemas de conjuntos de apoyo.

---

**Entrada:** $O_{new}$,$MA$

1: **para todo** $c_i$ en $MA$ **hacer**
2: $\quad \Phi_i(O_{new}) = \Phi_i^{'}(O_{new}) = 0$
3: $\quad \Omega_i^A = obtenerTestoresTipicos(MAbin(i))$
4: $\quad \Omega_i^A = seleccionarTestoresConAdaBoost(MAbin(i))$

5: **para todo** $\Omega_j^i$ en $\Omega_i^A$ **hacer**
6: **para todo** $O_k$ en $MAbin(i)$ **hacer**
7: **si** $y_k = c_i$ **entonces**
8: $voto(i)(j)(k) = 1 - HOEM(Onew, O_k)$
9: **si no**
10: $voto'(i)(j)(k) = 1 - HOEM(Onew, O_k)$
11: **fin si**
12: **fin para**
13: $\Gamma_j^i(O_{new}) =$ *obtenerPromedioDeLos5Mayores*$(voto(i)(j))$
14: $\Gamma_j'^i(O_{new}) =$ *obtenerPromedioDeLos5Mayores*$(voto'(i)(j))$
15: $\Phi_i(O_{new}) = \alpha_j^i * \Phi_i(O_{new}) + \Gamma_j^i(O_{new})$
16: $\Phi_i'(O_{new}) = \alpha_j^i * \Phi_i'(O_{new}) + \Gamma_j'^i(O_{new})$
17: **fin para**
18: $\Phi_i(O_{new}) = \Phi_i(O_{new})/|\Omega_i^A|$
19: $\Phi_i'(O_{new}) = \Phi_i'(O_{new})/|\Omega_i^A|$
20: $\lambda_i(O_{new}) = \Phi_i(O_{new}) + (1 - \Phi_i'(O_{new}))$
21: **fin para**
22: $win =$*argmax*$(\lambda_i(O_{new}))$ para $i = 1, \ldots, m$
23: **retornar** $c_{win}$

---

Generalizando la idea para cualquier algoritmo basado en AlVot PC, lo que se propone es seleccionar y ponderar los conjuntos de apoyo en cada sistema, que, en el contexto del algoritmo propuesto son los testores por clase. En adelante, cuando se utilice AdaBoost con este propósito en AlVot PC se denota concatenando el signo "+", es decir, AlVot PC+.

## 3.4. Criterios de comparación para la obtención de testores

Como se ha mencionado, los algorimos más eficientes para encontrar los testores asociados a una $MA$, reciben como parámetro una matriz binaria de comparación llamada $MC$ -que puede ser de diferencias, llamada $MD$-. Para obtener $MD$ se comparan -característica a característica- cada par de objetos de distinta clase en $MA$. La característica $x_k$ del objeto $O_s$ puede tener un valor nominal, numérico o incluso ser ausente, esto debe tomarse en cuenta para definir los criterios de comparación que permitan determinar si dos valores de $x_k$ se consideran iguales o diferentes.

Al comparar los valores $x_k^s$ y $x_k^p$, si alguno de ellos o ambos son faltantes,

es decir, no se conoce su valor, entonces se asume que son diferentes. Cuando $x_k$ sólo puede tomar valores nominales, basta con determinar si son iguales o no. A diferecia de comparar valores faltantes o nominales, el caso en que $x_k$ toma valores en un intervalo numérico no es tan trivial, por ejemplo, la estatura dada en metros de una muestra de mujeres adultas en el intervalo [1.42,1.85] $\in \Re$ ¿Se considera igual o diferente 1.65 de 1.66? y ¿1.50 de 1.82?

Dada una característica $x_k$ en $MA$ que toma valores numéricos, es de esperarse que dichos valores sean similares para objetos que pertenecen a la misma clase y aquellos valores que provienen de distintas clases sean muy diferentes. Bajo esta idea, al definir la Ecuación [3.1] como función de comparación entre los valores $x_k^p$ y $x_k^s$, se espera que esta tome valores cercanos a cero si $O_p$ y $O_s$ son de la misma clase y cercanos a uno si no lo son.

$$rn\_diff(x_k^p, x_k^s) = \frac{|x_p^i - x_p^j|}{rango(x_p)} \tag{3.1}$$

Al comparar todos los posibles valores de $x_k$ de una $MA$ con $m$ clases, se obtiene un conjunto de valores que se propone se agrupen de acuerdo a las clases, esto es, si $x_k^p$ y $x_k^s$ provienen de objetos de las clases $c_i$ y $c_j$ respectivamente (con $i \leq j$), entonces, $rn\_diff(x_k^p, x_k^s)$ pertenece a $D_{ij}$. Como resultado de dicho agrupamiento se tienen $n = \frac{m(m+1)}{2}$ grupos de valores. Cada $D_{ij}$ se asume tienen una distribución normal con media $\mu_{ij}$ y desviación estándar $\sigma_{ij}$. En la Figura 3.2 se muestra dicho agrupamiento para $m = 3$.

Una vez que se conocen las medias y desviaciones estándar de cada $D_{ij}$, se puede binarizar el resultado de $rn\_diff(x_k^p, x_k^s)$ con el objetivo de determinar si $x_k^p$ y $x_k^s$ se consideran diferentes o iguales. La propuesta para dicha binarización consiste en 3 pasos, el primero es asignar a $r_k$ el resultado de evaluar $rn\_diff(x_k^p, x_k^s)$, después se calculan las probabilidades de haber sido generadas por las distribuciones de $D_{ij}$, únicamente de las clases que provienen los valores $x_k^p$ y $x_k^s$. Por ejemplo, si $x_k^p$ proviene de un objeto de la clase $c_2$ y $x_k^s$ de la clase $c_3$, entonces se calculan las probabilidades de haber sido generadas por las distribuciones de $D_{22}$,$D_{33}$ y $D_{23}$, y por último se consideran iguales si la mayor probabilidad es de $D_{22}$ o $D_{33}$ y diferentes si es de $D_{23}$, tal como se ilustra en la Figura 3.3.

Con esta propuesta para binarizar el resultado de $rn\_diff(x_k^p, x_k^s)$, se puede definir un criterio de comparación boleano para comparar características numéricas.

A continuación se listan los criterios de comparación que permiten obtener $MD$:

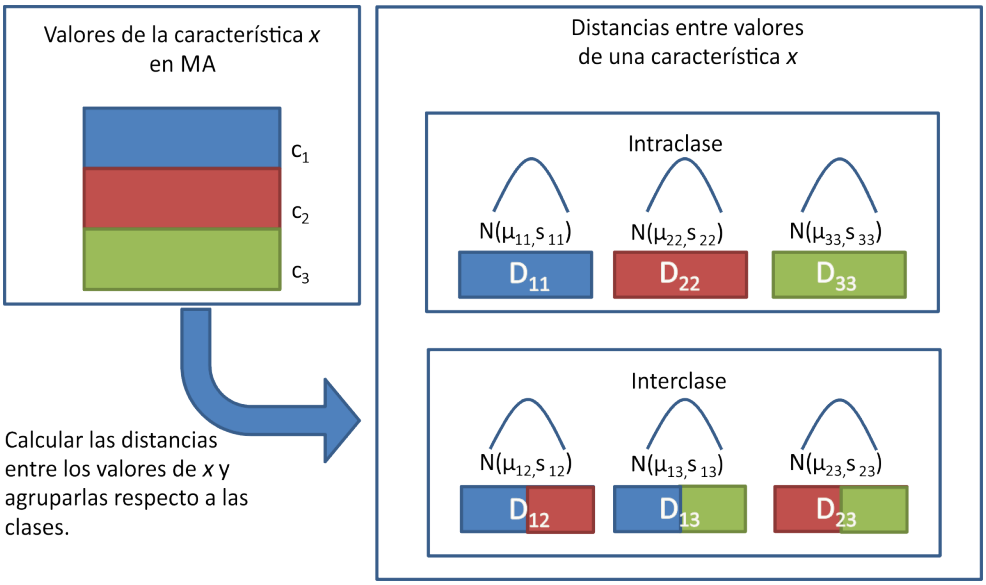


**Figura 3.2:** Esquema del cálculo de las distribuciones normales al comparar valores de una característica numérica $x$

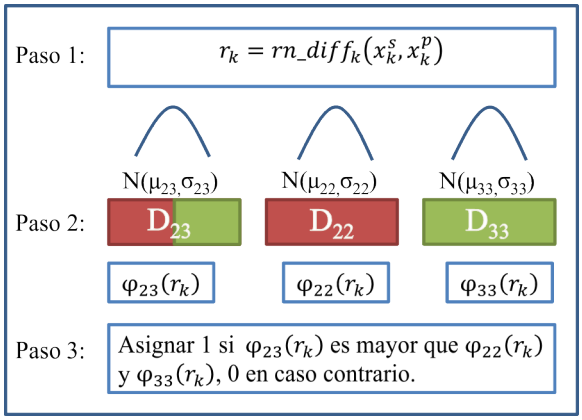


**Figura 3.3:** Ejemplo de los pasos para definir un criterio de comparación booleano para la la característica numérica $x_k$

- **Si $x_k$ es nominal.** $\phi_k(x_k^p, x_k^s) = \begin{cases} 1 & si\ x_k^p\ o\ x_k^s\ son\ faltantes \\ 1 & si\ x_k^p \neq x_k^s \\ 0 & en\ otro\ caso. \end{cases}$

- **Si $x_k$ es numérica.** $\phi_k(x_k^p, x_k^s) = \begin{cases} 1 & si\ x_k^p\ o\ x_k^s\ son\ faltantes \\ 1 & si\ \varphi_{ij}(r_k) > \varphi_{tw}(r_k) \quad i < j\ y\ t = w \\ 0 & en\ otro\ caso. \end{cases}$

  donde $r_k = rn\_diff(x_k^p, x_k^s)$.

# Capítulo 4

# Pruebas y resultados

En esta sección se presentan las pruebas realizadas para evaluar la cantidad de características utilizadas por los algoritmos propuestos respecto a su análogo basado en AlVot clásico, además de las pruebas para evaluar su desempeño de clásificación que incluye comparaciones respecto a otros algoritmos utilizados frecuentemente en la literatura.

Tanto para evaluar la cantidad de características como el desempeño en la clasificación, se realizó un sistema que implementa los algoritmos propuestos y su anágo basado en AlVot clásico (ver Apéndice 5). El sistema se utilizó para efectuar pruebas experimentales con diversos conjuntos de datos de la Universidad de California Irvin disponibles en UCI Machine Learning [Frank and Asuncion, 2010], los cuales se describen brevemente en la Tabla 4.1.

**Cuadro 4.1:** Descripción de los conjuntos de datos utilizados en las pruebas.

| Conjunto de datos | Número de características | Número de clases | Número de objetos | Tipo de características | ¿Datos faltantes? |
|---|---|---|---|---|---|
| Zoo | 16 | 7 | 101 | Nominales | No |
| Wine | 13 | 3 | 178 | Numéricas | No |
| Iris | 4 | 3 | 150 | Numéricas | No |
| Yeast | 8 | 10 | 1484 | Numéricas | No |
| Vote | 16 | 2 | 435 | Nominales | Sí |
| Diabetes | 8 | 2 | 768 | Numéricas | No |
| Credit | 20 | 2 | 1000 | Mezcladas | No |
| Glass | 9 | 6 | 214 | Numéricas | No |
| Vehicle | 18 | 4 | 846 | Numéricas | No |
| Anneal | 38 | 5 | 898 | Mezcladas | Sí |

Para cada conjunto se realizó validación cruzada estratificada con 10 pliegues. Dicha técnica divide cada conjunto en 10 partes iguales, clasifica en el paso $i$ la $i-$ésima parte y utiliza el resto como muestra de aprendizaje para $i = 1, \ldots, 10$.

## 4.1. Cantidad de características

Anteriormente se ha hecho mensión de que al definir un único sistema de conjuntos de apoyo para todas las clases (como en AlVot clásico), pueden existir características innecesarias para distinguir objetos de una clase específica. Por esta razón se espera que los algoritmos propuestos basados en AlVot PC y AlVot PC (con un sistema de conjuntos de apoyo para cada clase) utilicen menos características que su análago basado en AlVot clásico. En la Figura 4.1 se muestran los resultados de realizar validación cruzada con 10 folds en cada conjunto de datos y se reporta la moda del número de características utilizadas por los conjuntos de apoyo. Es importante notar que AlVot únicamente tiene un sistema de conjuntos de apoyo para todas las clases y por lo tanto la moda del número de características utilizadas en cada clase es la misma.

Los resultados revelan que los algoritmos propuestos, en cada conjunto de datos de más de 2 clases, existe al menos una clase de objetos que son diferenciables del resto al describirlos con menos características que las definidas por el algoritmo basado en AlVot clásico. Por ejemplo, en la clase $c_1$ del conjunto de datos *Zoo* (ver Figura a) 4.1 ), el algoritmo basado en AlVot clásico utiliza 8 caracteríscas y los basados en AlVot PC y AlVot PC+ utilizan 6 y 4 respectivamente.

En el caso de problemas de dos clases, era de esperarse que los algoritmos basados en AlVot clásico y AlVot PC utilizaran el mismo número de características, ya que al realizar la binarización de las clases con la técnica *una clase contra todas*, el resultado son dos problemas binarios equivalantes al original y si los conjuntos de apoyo son calculados de la misma manera, estos tendrán los mismos conjuntos de apoyo. Sin embargo, el algoritmo basado en AlVot PC+ permite seleccionar conjuntos de apoyo distintos para ambas clases, que en los resultados de la Figura e),f) y g) 4.1 , la moda del número de caracteríscas se manetiene.

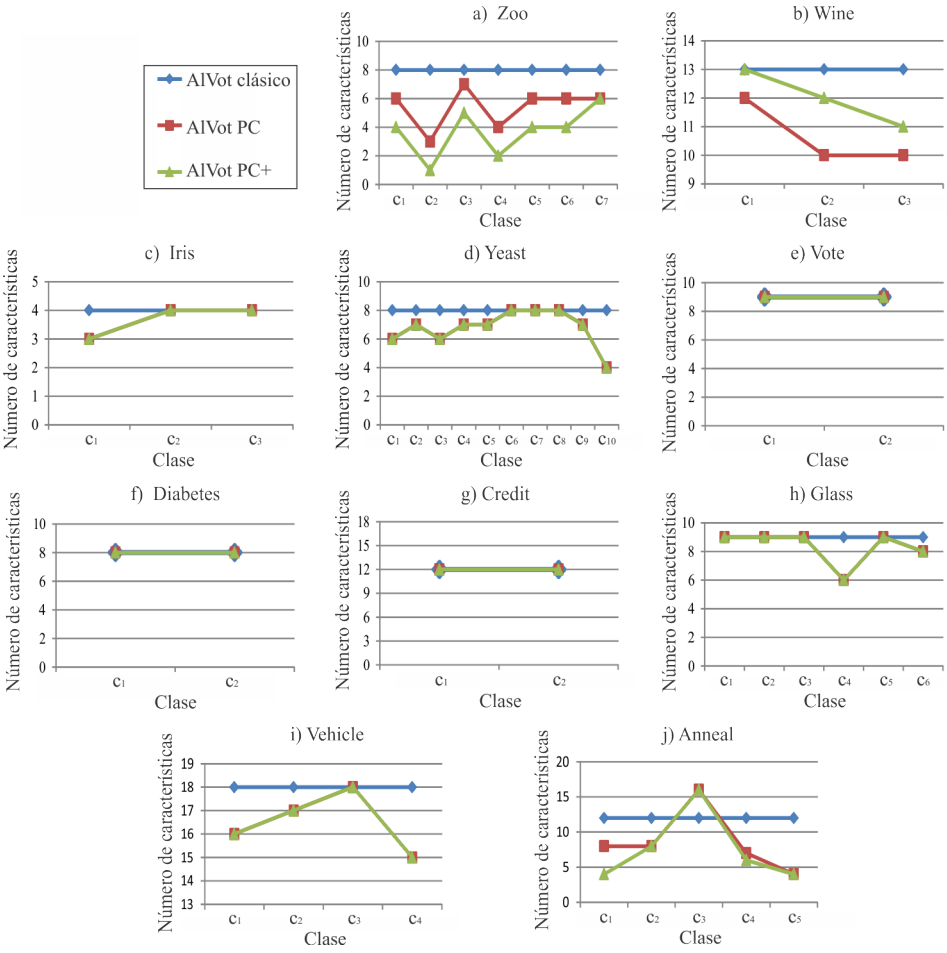


**Figura 4.1:** Moda de la cantidad de características utilizadas por los conjuntos de apoyo en los algoritmos propuestos basados en AlVot PC, AlVot PC+ y su análago basado en AlVot clásico.

## 4.2. Desempeño de clasificación

El objetivo de comparar el desempeño de clasificación de los algoritmos propuestos con su análogo basado en AlVot clásico es verificar que realmente existe una diferencia significativa al utilizar un sistema de conjuntos de apoyo para cada clase y no uno para todas. Además, es importante que los algoritmos propuestos sean comparados con otros usados frecuentemente en la literatura, con el objetivo de definir su competividad. En ambas comparaciones se reporta el promedio del porcentaje de clasificación al relizar 5 veces validación cruzada con 10 pliegues.

### 4.2.1. Comparación con AlVot clásico.

En la Tabla 4.2 se muestran los resultados de realizar el proceso de evaluación descrito al inicio de esta sección para los algoritmos basados en AlVot PC, AlVot PC+ y de su análogo AlVot clásico. Se muestran remarcados los porcentajes de clasificación más altos para cada conjunto de datos. Es importante notar que las columnas AlVot PC y AlVot PC+ hacen referecia a los algoritmos propuestos y no a los modelos, de igual manera para el algoritmo análogo basado en AlVot clásico.

**Cuadro 4.2:** Promedio del porcenteje de clasificación al evaluar los algoritmos propuestos basados en AlVot PC, AlVot PC+ y su análago basado en AlVot clásico.

| Conjunto de datos | AlVot clásico (%) | AlVot PC (%) | AlVot PC+ (%) |
|---|---|---|---|
| Zoo | 93.861 | 94.059 | **96.238** |
| Wine | 97.303 | **97.865** | 97.753 |
| Iris | 95.467 | **95.733** | **95.733** |
| Yeast | **59.582** | 59.070 | 59.057 |
| Vote | 93.885 | 93.885 | **94.713** |
| Diabetes | **73.542** | **73.542** | **73.542** |
| Credit | **71.900** | **71.900** | 71.880 |
| Glass | 69.092 | 70.364 | **70.497** |
| Vehicle | 70.501 | **70.968** | **70.968** |
| Anneal | 97.884 | 98.018 | **98.151** |

Para deteriminar si existe diferencia significativa en los resultados de las comlumnas AlVot clásico, AlVot PC y AlVot PC+, se hace uso de una prueba de significancia estadística conocida como *Wilcoxon Signed-Ranks test* (WSR) debido a que permite la comparación de dos o más clasificadores

sobre múltiples conjuntos de datos [Demšar, 2006]. Las pruebas se realizaron con un nivel de significacia de 0.1 y al comparar AlVot clásico con AlVot PC, revelan que a pesar de que este últmo únicamente tiene un menor porcentaje de clasificación que AlVot clásico con el conjunto de datos *Yeast*, la diferencia no se considera significativa. Sin embargo, al comparar AlVot PC+ contra AlVot clásico, si existe diferencia significativa a favor de AlVot PC+. Con esto, se puede afirmar con un nivel de confianza del 90 % que al repetir esta misma prueba con otros conjuntos de datos, AlVot PC+ tendrá un mejor desempeño de clasificación que AlVot clásico.

A simple vista, los porcentajes de clasificación obtenidos por AlVot PC+ pueden parecer muy cercanos a los de AlVot clásico, por esta razón es importante mencionar que dependiendo del problema de clasificación por resolver, es el impacto de un pequeño o gran incremento en el porcentaje de clasificación. Por ejemplo, en problemas en que la integridad humana está de por medio, tales como el clasificar a una persona como enferma o no, culpable o inocente, rechazada o aceptada, etc., el incremento de un punto porcentual implica gran impacto.

### 4.2.2. Comparación con otros algoritmos de la literatura.

Al evaluar el desempeño de clasificación de algoritmos ampliamente utilizados en la literatura tales como K-NN, C4.5 y Naive Bayes (NB) y al compararlos con el desempeño de los algoritmos propuestos, se aprecia en la Tabla 4.3 que tanto el basado en AlVot PC como en AlVot PC+ son competitivos, ya que ambos, junto con el algoritmo C4.5 obtienen los mejores porcentajes de clasificación (remarcados en la tabla).

Es importante mensionar que el desempeño de clasificación de un algoritmo en particular, depende en gran medida del problema que se esté resolviendo. Por ejemplo, en la Tabla 4.3 el algoritmo Naive Bayes tiene los mejores porcentajes de clasificación para los conjuntos de datos *Diabetes* y *Credit*, sin embargo, para los conjuntos *Glass*, *Vehicle* y *Anneal* tiene los más bajos porcentajes. Para dar una idea del comportamiento general de los algoritmos evaluados, en la Figura 4.2 se muestra el promedio de los porcentajes de clasificación de cada uno de ellos, es decir, los datos de cada columna en la Tabla 4.3 fueron promediados. La figura se aprecia que el algoritmo basado en AlVot PC+ tiene el mejor desempeño respecto al resto de clasificadores.

**Cuadro 4.3:** Promedio del porcenteje de clasificación al evaluar los algoritmos K-NN (con K=3 y K=5), C4.5, Naive Bayes y los propuestos basados en AlVot PC, AlVot PC+.

| Conjunto de datos | 3-NN (%) | 5-NN (%) | C4.5 (%) | NB (%) | AlVot PC (%) | AlVot PC+ (%) |
|---|---|---|---|---|---|---|
| Zoo | 92.475 | 94.653 | 92.277 | 95.050 | 94.059 | **96.238** |
| Wine | 96.067 | 96.067 | 92.809 | 97.191 | **97.865** | 97.753 |
| Iris | 95.333 | **95.733** | 94.667 | 95.600 | **95.733** | **95.733** |
| Yeast | 55.431 | 56.968 | 56.226 | 57.911 | **59.070** | 59.057 |
| Vote | 93.057 | 93.241 | **96.460** | 90.161 | 93.885 | 94.713 |
| Diabetes | 72.943 | 73.698 | 74.583 | **75.469** | 73.542 | 73.542 |
| Credit | 72.340 | 73.000 | 70.940 | **74.700** | 71.900 | 71.880 |
| Glass | 70.467 | 66.729 | 68.505 | 48.131 | 70.364 | **70.497** |
| Vehicle | 70.213 | 70.307 | **72.908** | 44.846 | 70.968 | 70.968 |
| Anneal | 97.305 | 97.372 | **98.708** | 86.548 | 98.018 | 98.151 |

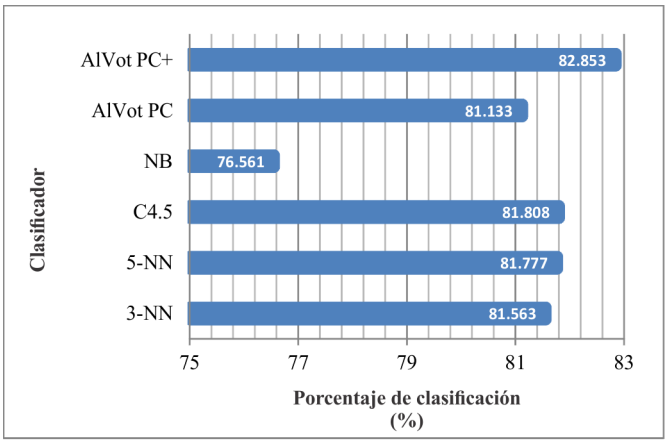


**Figura 4.2:** Promedio de los porcentajes de clasificación al realizar 5 veces validación cruzada con 10 folds con los algoritmos K-NN (con K=3 y K=5), C4.5, Naive Bayes y los propuestos basados en AlVot PC, AlVot PC+.

# Capítulo 5

# Conclusiones y trabajo futuro

En este trabajo de tesis se modificó AlVot clásico para considerar un sistema de conjuntos de apoyo para cada clase, esto con el objetivo de que cada conjunto de apoyo de una clase específica, contega las características que perimiten diferenciar a objetos su clase asociada respecto al resto. Al modelo resultante se le denominó AlVot PC. La hipótesis es que los algoritmos basados en el modelo propuesto, mejoran el desempeño de clasificación respecto a aquellos basados en el modelo original. Para probar esto de manera experimental, se propuso un algoritmo basado en AlVot PC que utiliza a los testores por clase como sistemas de conjuntos de apoyo. Este algoritmo fué implementado junto con su análago basado en AlVot clásico y evaluado con 10 conjuntos de datos. En las pruebas realizadas se mustró que sí mejora el porcentaje de clasificación aunque no de manera significativa.

Por otra parte, cada conjunto de apoyo puede tener un mayor peso en la clasificación de nuevos objetos y si su peso es muy bajo, puede resultar innecesario, por esta razón se propuso utilizar el algoritmo AdaBoost para ponderar y seleccionar a cada conjunto apoyo. Cuando los algoritmos utilizan dicha forma de ponderación y selección, se dice que están basados en AlVot PC+, entonces, el algoritmo propuesto fué modificado para estar basado en AlVot PC+, se implementó y se evaluó su desempeño de clasificación. Los resultados también muestran una mejora respecto al desempeño de su análogo basado en AlVot clásico, pero ahora de manera significativa.

De lo anterior se puede afirmar que considerar un sistema de conjuntos de apoyo para cada una de las clases en AlVot, permite mejorar el desempeño en la clasificación. Esto da indicios de que las características innecesarias al considerar un único sistema, generan ruido en la clasificación. En las pruebas experimentales también se evaluó la cantidad de caracteríticas utilizadas por

en los conjuntos de apoyo y los resultados muestran que para problemas de mas dos clases, los algoritmos propuestos usan menos características que los basados en AlVot clásico.

Una de las principales ventajas de los algoritmos propuestos es que permiten trabajar con objetos descritos por características numéricas, nominales o mezcladas, e incluso con datos faltantes. En los resultados de las pruebas experimentales se aprecia también que ambos algoritmos son competitivos y pueden resultar una buena alternativa para los clasificadores más utilizados en la literatura.

En este proyecto de tesis se utilizaron a los testores típicos por clase para definir cada sistema de conjuntos de apoyo, sin embargo, como trabajo futuro puede definirse otro algoritmo basado en AlVot PC que determine a cada conjunto de apoyo de otra manera, por ejemplo, utilizando otros selectores de características [Liu and Motoda, 2008] pero considerando los problemas binarios de aplicar el método *una clase contra todas*. Al igual, para seleccionar y ponderar cada conjunto de apoyo se puede probar mediante otras técnicas, ya sea con otra técnica de ensamble de clasificadores o proponiendo una manera que no implique la intervención de otro clasificador.

# Apéndice A

# Sistema que implementa los algoritmos propuestos

Con el propósito de evaluar el desempeño de clasificación de los algoritmos propuestos basados en AlVot PC y en AlVot PC+, se implementó un sistema que incluye también el algoritmo análogo basado en AlVot clásico. El sistema se elaboró con el leguaje de programación *Java*, haciendo uso de *NetBeans IDE* [Tulach and Jarda, 2011] y una librería hecha en *Java* llamada *Weka* [Hall et al., 2009], estas aplicaciones cuentan con licencia *GNU*. *NetBeans IDE* es un ambiente de desarrollo integrado para programadores que proporciona las herramientas necesarias para crear aplicaciones profesionales de escritorio, empresariales, web y móviles con la plataforma de Java, asi como con *C/C++*, *PHP*, *JavaScript* y *Groovy*. *Weka* tiene implementados diversos algoritmos para la minería de datos, incluyendo entre otros componentes, métodos de clasificación y clustering, herramientas para el preprocesamiento de los datos como lo son filtros para los atributos, además de distintos métodos para la selección de características.

Para que el sistema desarrollado permita evaluar el desempeño de clasificación de los algoritmos propuestos y de aquel basado en AlVot clásico, este debe debe cumplir con 5 requerimientos listados a continuación:

RF-1 Seleccionar la muestra de aprendizaje.

RF-2 Seleccionar el clasificador a evaluar.

RF-3 Ajustar la forma de evaluación.

RF-4 Evaluar el clasificador.

RF-5 Visualizar resultados.

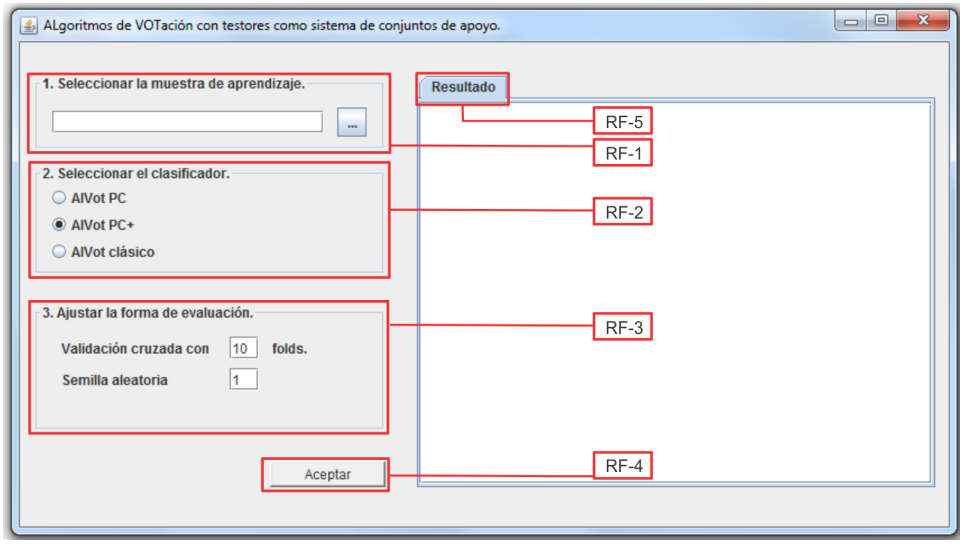


**Figura A.1:** Pasos para definir un criterio de comparación booleano para la la característica numérica $x_k$.

En la Figura A.1 se presenta la interfaz gráfica del sistema enmarcando en color rojo los ojetos que permiten la ejecución de cada requerimiento.

Para mostrar cómo el sistema evalúa el desmpeño de clasificación, a continuación se describen los pasos a seguir para evaluar el algoritmo basado en AlVot PC+ con el conjunto de datos *Zoo* descrito en la Tabla 4.1 con base en la interfáz mostrada en la Figura A.1:

Paso 1. Dar clic en el botón "..." ubicado en el panel "1. Seleccionar muestra de aprendizaje"desplegandose la ventana para seleccionar el archivo que contiene los datos del conjunto *Zoo*.

Paso 2 En el panel "2. Seleccionar clasificador" se selecciona "AlVot PC+".

Paso 3 En el panel "3. Ajustar forma de evaluación" escribir el número de pliegues y la semilla aleatoria con que se realizará la validación cruzada, el default es de 10 y 1 respectivamente.

Paso 4 Dar clic en el botón "Aceptar".

Paso 5 Visualizar el rusultado de la evalueción en el area "Resultado".

En la Figura A.2 se muestra la respuesta del sistema al realizar estos pasos.

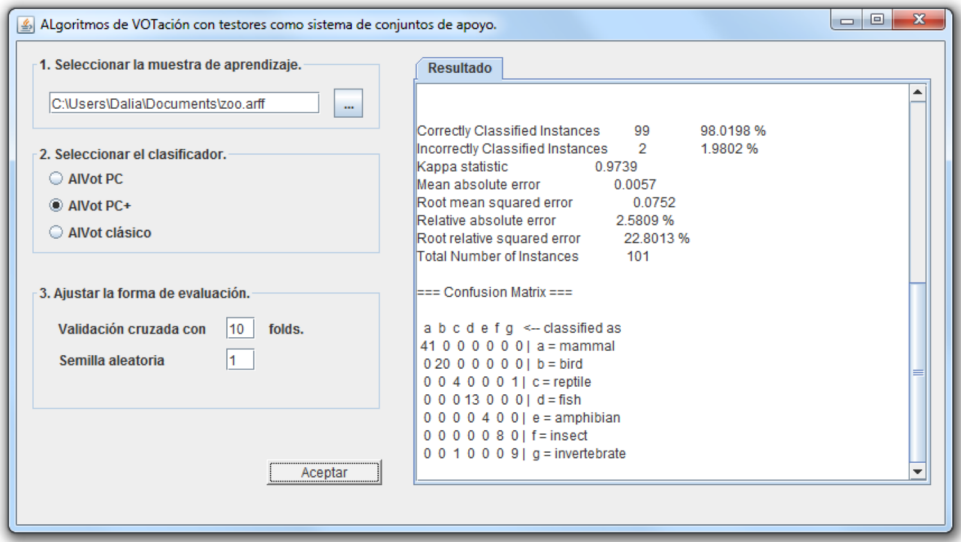


**Figura A.2:** Pasos para definir un criterio de comparación booleano para la la característica numérica $x_k$.

Es importante mencionar que dado a que se utilizó la librería *Weka*, al seleccionar la muestra de aprendizaje, los datos deben cumplir con el formato *Arff* descrito en [Hall et al., 2009].

# Bibliografía

[A. et al., 2013] A., R., P.D., M., and S., M. (2013). Methods for pattern selection, class-specific feature selection and classification for automated learning. *Elsevier*, 41:113–129.

[Baggenstoss, 1998] Baggenstoss, P. (1998). Class-specific feature sets in classification. In *ISIC/CIRA/ISAS Joint Conference*, pages 413–416. IEEE.

[Baggenstoss, 1999] Baggenstoss, P. (1999). Class-specific feature sets in classification. In *Transactions on signal processing*, volume 47, pages 3428–3432. IEEE.

[Chegis and Yablonsky, 1958] Chegis, I. and Yablonsky, S. (1958). Logical methods for controlling electrical circuits. *Trudy Matematicheskogo Instituta Steklova*, 51:270–360.

[Demšar, 2006] Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets.

[Frank and Asuncion, 2010] Frank, A. and Asuncion, A. (2010). Uci machine learning repository.

[Freund and Schapire, 1995] Freund, Y. and Schapire, R. (1995). A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, pages 23–37. Springer.

[Hall et al., 2009] Hall, M., Eibe, F., Geoffrey, H., Bernhard, P., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: An update. In *SIGKDD Explorations*, volume 11, pages 10–18.

[J. Martínez-Trinidad and Contreras-Arévalo, 2000] J. Martínez-Trinidad, M. V.-S. and Contreras-Arévalo, E. (2000). Discovering differences in patients with uveitis through typical testors by class. In Zighed, D., Komo-

rowski, J., and Żytkow, J., editors, *Principles of Data Mining and Knowledge Discovery*, volume 1910 of *Lecture Notes in Computer Science*, pages 524–529. Springer Berlin Heidelberg.

[Lazo-Cortés et al., 1998] Lazo-Cortés, M., de-la Peña, M. D., and Quintana-Gómez, T. (1998). Testor by class: an application to character recognition. *Expert Systems with Applications*, 37(5):229–236.

[Lazo-Cortes et al., 2001] Lazo-Cortes, M., Ruiz-Shulcloper, J., and Alba-Cabrera, E. (2001). An overview of the evolution of the concept of testor. *Pattern recognition*, 34(4):753–762.

[Liu and Motoda, 2008] Liu, H. and Motoda, H. (2008). *Computational methods of feature selection*. 1ra edition.

[Pineda-Bautista et al., 2011] Pineda-Bautista, B., Carrasco-Ochoa, J., and Martínez-Trinidad, J. (2011). General framework for class-specific feature selection. *Expert Systems with Applications*, 38(8):10018–10024.

[Ruiz-Shulcloperand and Abidi, 2002] Ruiz-Shulcloperand, J. and Abidi, A. M. (2002). Logical combinatorial pattern recognition: A review.

[Tulach and Jarda, 2011] Tulach and Jarda (2011). Netbeans integrated development environment.

[Zhuravlev and Nikiforov, 1971] Zhuravlev, Y. and Nikiforov, V. (1971). Algorithms for recognition based on calculation of evaluations. *Kibernetilca (in Russian)*, (3):1–11.