MÉTODOS PARA MEJORAR LA CALIDAD DE UN CONJUNTO DE DATOS EN DESCUBRIMIENTO DE CONOCIMIENTO

Edgar Acuña y Luis Daza

Universidad de Puerto Rico
Recinto Universitario de Mayagüez

Contenido

I-INTRODUCCION

Objetivo

Desarrollar medidas de complejidad de los conjuntos de datos en problemas de clasificación supervisada e implementar técnicas eficientes de detección de ruido, selección de variables e instancias para optimizar el rendimiento de los algoritmos de la minería de datos.

Objetivos

Objetivos específicos:

1. Construir medidas eficientes de la complejidad de los conjuntos de datos, que permitan describir el comportamiento de los algoritmos en problemas de clasificación supervisada.

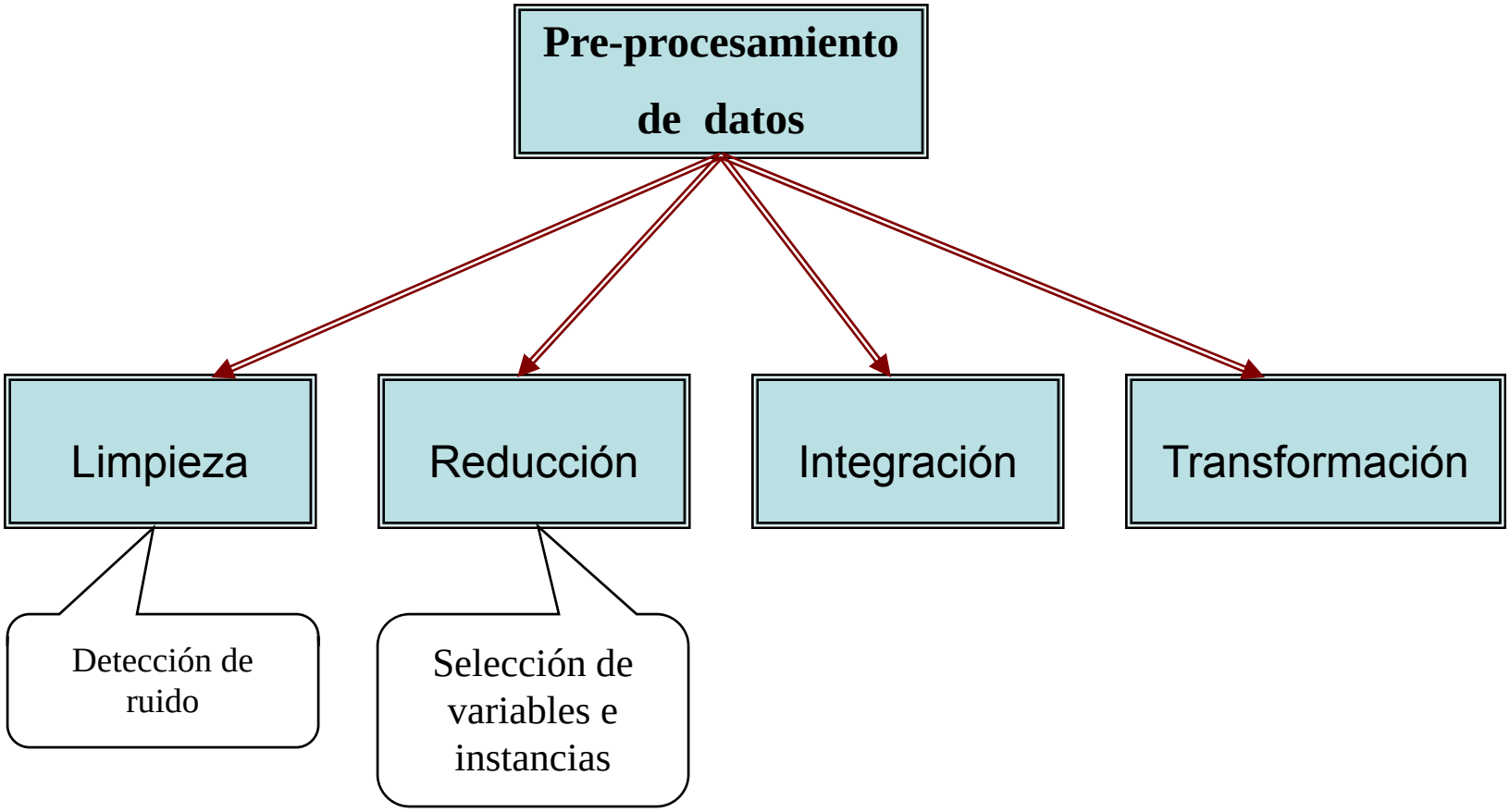
2. Desarrollar e implementar un algoritmo para identificar y eliminar el ruido en el conjunto de entrenamiento con la finalidad de mejorar la precisión de los clasificadores.

3. Elaborar métodos eficientes de selección de variables basados en la complejidad del problema de clasificación.

4. Mejorar la escalabilidad de los algoritmos de selección de instancias, que aumente su capacidad de reducir el conjunto de entrenamiento con un menor costo computacional.

5. Combinar la detección y eliminación de ruido, con la selección de variables e instancias con la finalidad de mejorar el rendimiento de los clasificadores.

Pre-procesamiento de los datos



Referencias: Pyle, D. 1999, Dasu, T and Johnson, T. 2003

II. Complejidad de los datos

Varios estudios han mostrado que el comportamiento empírico de los clasificadores está fuertemente relacionado a los datos disponibles. *A priori*, es muy difícil determinar que clasificador tendrá un mejor rendimiento, dado un problema específico.

Factores que contribuyen a la complejidad de un problema de clasificación:

a) **La ambigüedad de la clase**.

b) **Tamaño y dimensión del conjunto entrenamiento**.

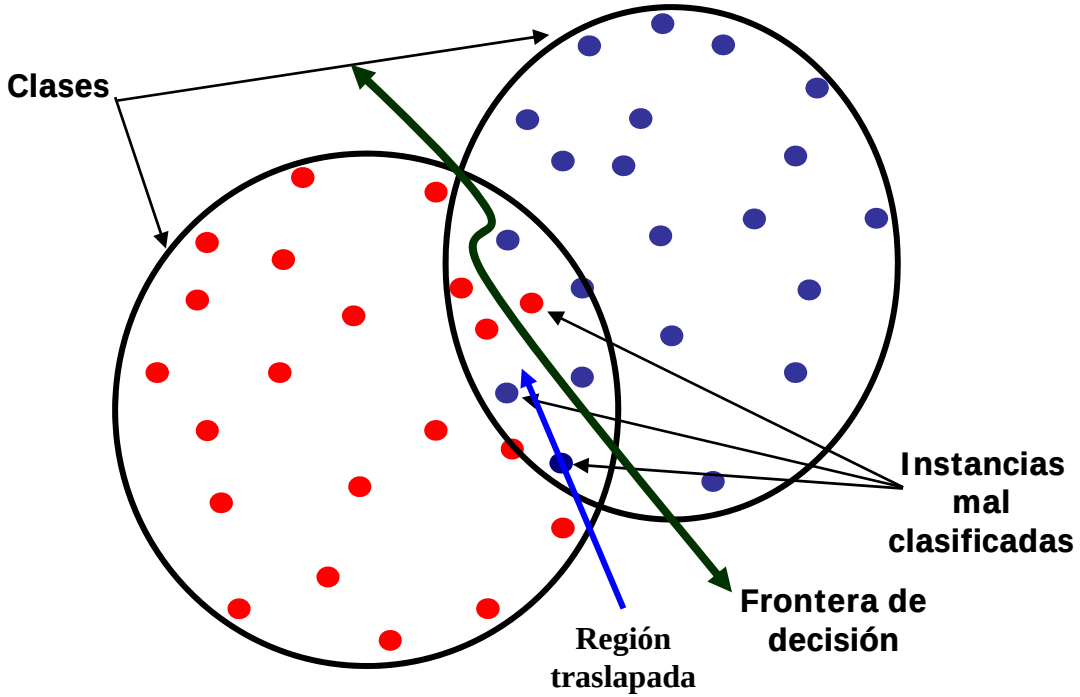
c) **Complejidad de la frontera de decisión.**

Medidas para la caracterización de datos

Una forma simple de cuantificar la complejidad de un problema de clasificación es calcular la tasa de error de clasificación para un clasificador elegido. Sin embargo, si lo que se quiere es estudiar el comportamiento de los clasificadores, se tienen que usar otras medidas que sean independientes de la elección del clasificador pero capaces de discernir su competencia.

Existen diferentes medidas de complejidad de los datos, estas pueden medir la región de traslapo de dos o mas clases, la separabilidad de las clases y/o describir la geometría o las formas de las clases.

Medidas para la caracterización de datos



Medidas para la caracterización de datos

Razón discriminante de Fisher's (F1)

La versión más simple de esta medida es muy conocida y estima cuan separables están dos clases mediante una variable específica (Ho, 2001, 2002)

$$F_1 = \frac{\left(\mu_1 - \mu_2\right)}{\sigma_1^2 + \sigma_2^2}$$

La generalización para ***g*** clases y que también considera todas las variables, es la siguiente (R. Mollineda, J. *Sanchez y J. Sotoca*, 2005 ; Sotoca, J., Sanchez, J., y Mollineda, R., 2005):

$$F1_{gen} = \frac{\sum_{i=1}^{g} n_i \delta\left(\mu, \mu_i\right)}{\sum_{i=1}^{g}\sum_{j=1}^{n_i} \delta\left(x_j^i, \mu_i\right)}$$

Donde δ es una metrica.

Medidas para la caracterización de datos

Volumen de la región de traslapo (*overlap*, F2)

Esta medida calcula, para cada variable f_k (k=1,..d) , la longitud del traslapo normalizado por la longitud del rango total en que todos los valores de ambas clases están distribuidos (Ho, 2001, 2002)

$$F2 = \prod_k \frac{\min\max_k - \max\min_k}{\max\max_k - \min\min_k}$$

Donde $\min\max_k = \min\{\max(f_k, c_1), \max(f_k, c_2)\}$ y similarmente los otros

Una generalización muy simple de la medida F2 para el problema del C-clases puede obtenerse sumando la medida para todos los posibles pares de clases (Mollineda, *et al*., 2005)

$$F2_{gen} = \sum_{(c_i, c_j)} \prod_k \frac{\min\max_k - \max\min_k}{\max\max_k - \min\min_k}$$

Medidas para la caracterización de datos

Fracción de puntos en la frontera (F3)

Este método se basa en el uso de *Minimum Spanning Tree* (MST) propuesto por Friedman y Rafsky (1979). El método construye un MST que conecta todas las instancias en el conjunto de entrenamiento a su vecino más cercano, sin considerar la clase a la que pertenece. Luego se procede a contar el número de instancias conectadas a una instancia de la clase contraria por una arista en el MST. Se considera que estos puntos están cerca de la frontera de la clase. La fracción de las instancias restantes sobre el número total de instancias en el conjunto de entrenamiento se usa como una medida de complejidad (F3) (Jain *et al*., 2002).

Medidas de la calidad de las instancias (Daza, 2007)

Calidad de las instancias con respecto a los centroides (Q)

Dado un conjunto de entrenamiento T con n instancias, para cada instancia x_i en T que pertenece a una de las g clases $\{G_1, G_2, \cdots, G_g\}$, se calcula la siguiente medida de calidad de la instancia:

$$Q_i = \frac{r_i - d_i}{\max(d_i, r_i)}$$

Donde: d_i es la distancia de la i-ésima instancia (x_i) al centroide de su misma clase. r_i es la distancia mínima de la i-ésima instancia a los centroides de las clases opuestas. Normalizada por el máximo de d y r.

Medidas de la calidad de las instancias (Daza, 2007)

Calidad de las instancias con respecto a su vecino mas cercano (QNN)

Dado un conjunto de entrenamiento T con n instancias, para cada instancia I_i *en E que pertenece a una de las g clases {G1,….Gg}, se calcula la siguiente medida de calidad de la instancia:*

$$QNN_i = \frac{\tilde{r}_i - \tilde{d}_i}{\max(\tilde{r}_i, \tilde{d}_i)}$$

donde: $\tilde{d}_i$ es la distancia métrica de la *i-ésima instancia Ii a su vecino más cercano en su misma clase;* $\tilde{r}_i$ *es la distancia métrica mínima de la i-ésima instancia a sus vecinos más cercanos de cada una de las clases opuestas, normalizada por el máximo de* $\tilde{d}_i$ *y* $\tilde{r}_i$.

Medidas de complejidad basadas en la calidad de las instancias

Para usar las medidas de calidad de las instancias en determinar la complejidad de un conjunto de datos procedemos como sigue:

1. Para cada instancia del conjunto de entrenamiento, se calcula la medida de calidad Q (o QNN).
2. Se procede a contar las instancias que tienen una medida de calidad menor que cero (negativa). La medida reporta el % de instancias con "mala" calidad o ruido, respecto al total de instancias en el conjunto de entrenamiento (n):

$$Q_{Ind} = \frac{count\{Q_i < 0\}}{n}$$

Medida propuesta de traslapo (OVER)

Dado un conjunto de entrenamiento E con n instancias, para cada una de las dos clases $\{G_1, G_2\}$, se calcula la distancia promedio de cada instancia al centro de su clase:

$$R_{G_1} = \left(\frac{\sum_{x \in G_1} d\left(x, C_{G_1}\right)}{n_{G_1}} \right) \qquad R_{G_2} = \left(\frac{\sum_{x \in G_2} d\left(x, C_{G_2}\right)}{n_{G_2}} \right)$$

Luego, la medida de separabilidad se calcula, dividiendo la suma de las distancias promedios por la distancia de los centroides entre las clases $\{G_1, G_2\}$, de esta forma:

$$Overlap = \frac{\left(\frac{\sum_{x \in G_1} d\left(x, C_{G_1}\right)}{n_{G_1}} \right) + \left(\frac{\sum_{x \in G_2} d\left(x, C_{G_2}\right)}{n_{G_2}} \right)}{d\left(C_{G_1}, C_{G_2}\right)}$$

Medida propuesta de traslapo (OVER)

Una generalización muy simple de la medida de traslapo para el problema de g-clases, se obtiene promediando la medida para todos los posibles pares de clases:

$$OVER = \sum_{(G_i, G_j)} \frac{\left(\frac{\sum_{x \in G_i} d\left(x, C_{G_i}\right)}{n_{G_i}} \right) + \left(\frac{\sum_{x \in G_j} d\left(x, C_{G_j}\right)}{n_{G_j}} \right)}{d\left(C_{G_j}, C_{G_j}\right)}$$

RESULTADOS

Tasas de error de clasificación usando validación cruzada para los diferentes conjuntos de datos

CONJUNTO	LDA	KNN	SVM	RPART	LOG
ABALONE	36.12	38.82	**34.15**	37.62	35.09
BALANCE	13.28	13.62	**9.57**	21.57	10.75
BREASTW	4.00	3.19	**3.05**	5.42	3.28
BUPA	32.03	36.26	**29.97**	31.68	31.19
CENSUS	17.51	24.89	**15.47**	15.91	16.28
DIABETES	22.99	30.48	23.79	25.96	**22.54**
IONOSFERA	14.62	15.61	**5.56**	12.39	16.32
IRIS	**2.00**	4.00	3.47	6.80	2.47
LANDSAT	16.07	**8.95**	10.08	18.66	16.74
LETTER	42.83	**4.77**	7.11	60.72	7.11
PENBASED	12.42	0.66	**0.51**	18.21	8.72
SEGMENT	8.53	**4.66**	5.54	8.17	4.92
SHUTTLE	5.61	0.17	**0.11**	0.53	3.14
SONAR	25.29	18.61	**16.01**	30.05	25.58
VEHICLE	22.16	35.01	23.33	32.29	**20.30**
WAVEFORM	13.93	23.33	13.90	26.63	**13.25**
Promedio	**18.09**	**16.44**	**12.60**	**22.04**	**14.85**

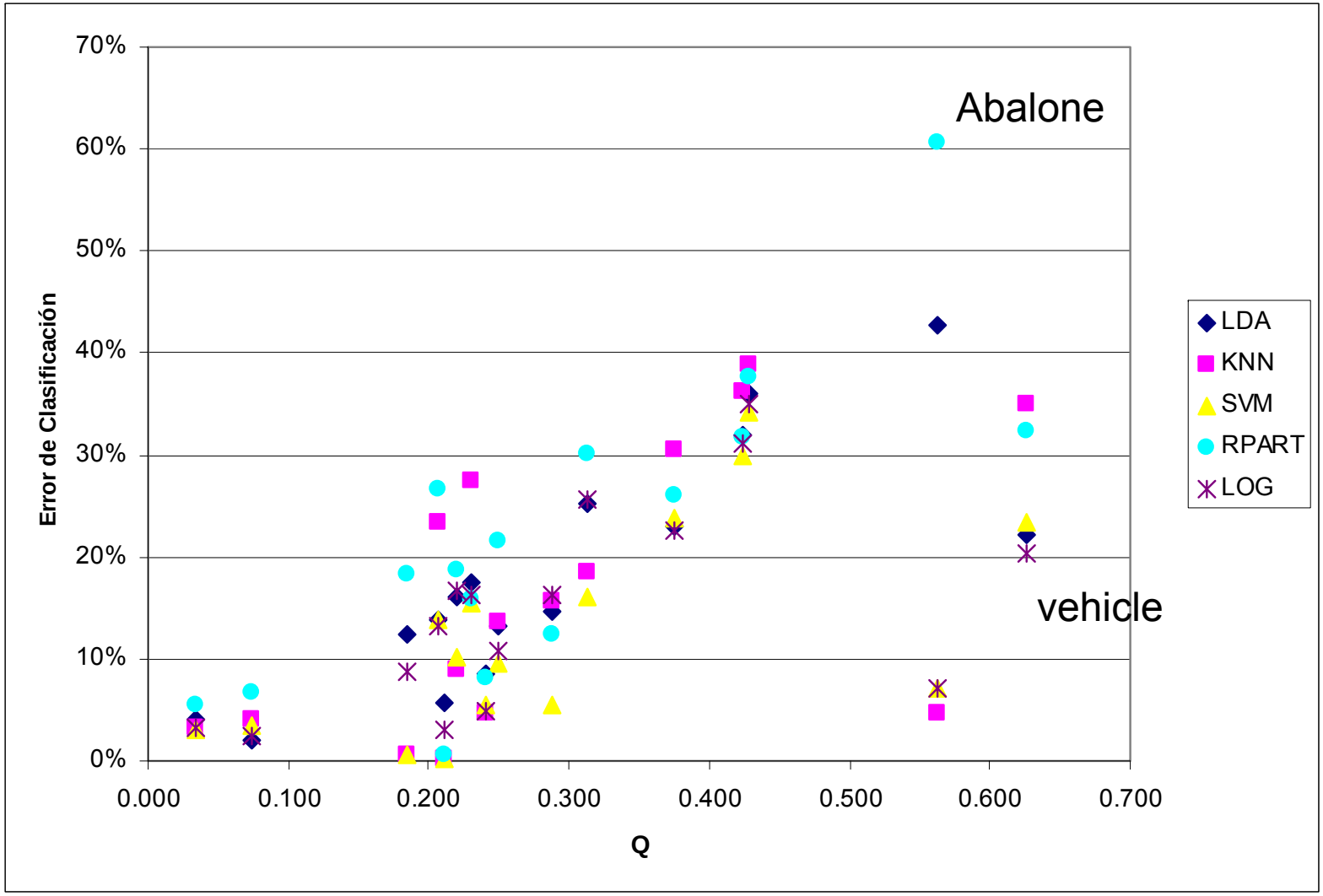
Medidas de complejidad para los diferentes conjuntos de datos

CONJUNTO	F1	F2	F3	Q	QNN	OVER	Qnorm
ABALONE	0.374	0.120	0.588	0.428	0.367	4.460	0.423
BALANCE	0.453	3.000	0.792	0.250	0.110	3.150	0.250
BREASTW	**1.353**	0.217	0.956	**0.034**	**0.028**	**0.760**	**0.035**
BUPA	0.165	0.073	0.623	0.423	0.354	6.200	0.400
CENSUS	0.194	0.212	0.726	0.230	0.200	3.917	0.230
DIABETES	0.217	0.252	0.680	0.375	0.275	4.540	0.266
IONOSFERA	0.235	**0.043**	0.872	0.288	0.134	4.061	0.282
IRIS	**2.665**	0.054	**0.960**	**0.073**	0.040	**0.532**	**0.067**
LANDSAT	**1.476**	**0.000**	0.910	0.220	0.085	**0.970**	0.219
LETTER	0.531	2.593	**0.961**	0.562	**0.036**	2.723	0.562
PENBASED	**1.161**	2.060	**0.994**	**0.185**	**0.006**	**1.232**	**0.185**
SEGMENT	**1.146**	**0.000**	**0.967**	0.240	**0.036**	**1.535**	**0.159**
SHUTTLE	0.595	0.089	**1.000**	**0.211**	**0.002**	1.941	0.214
SONAR	0.175	**0.000**	0.827	0.313	0.173	5.363	0.260
VEHICLE	0.505	0.169	0.656	0.626	0.329	5.602	0.547
WAVEFORM	0.516	**0.001**	0.768	**0.207**	0.192	2.289	**0.195**

Correlación entre las tasas de error de clasificación y las medidas de complejidad

	LDA	KNN	SVM	RPART	LOG
LDA	1				
KNN	0.565	1			
SVM	0.673	0.953	1		
RPART	0.930	0.436	0.531	1	
LOG	0.679	0.877	0.918	0.488	1
F1	-0.584	-0.601	-0.527	-0.443	-0.597
F2	0.216	-0.313	-0.272	0.39	-0.278
F3	-0.546	-0.988	-0.953	-0.428	-0.862
Q	0.826	0.585	0.618	0.798	0.555
QNN	0.581	0.992	0.970	0.447	0.900
OVER	0.653	0.836	0.790	0.499	0.819
Qnorm	0.856	0.536	0.570	0.83	0.528

Relación entre la tasa de error y la medida de complejidad Q



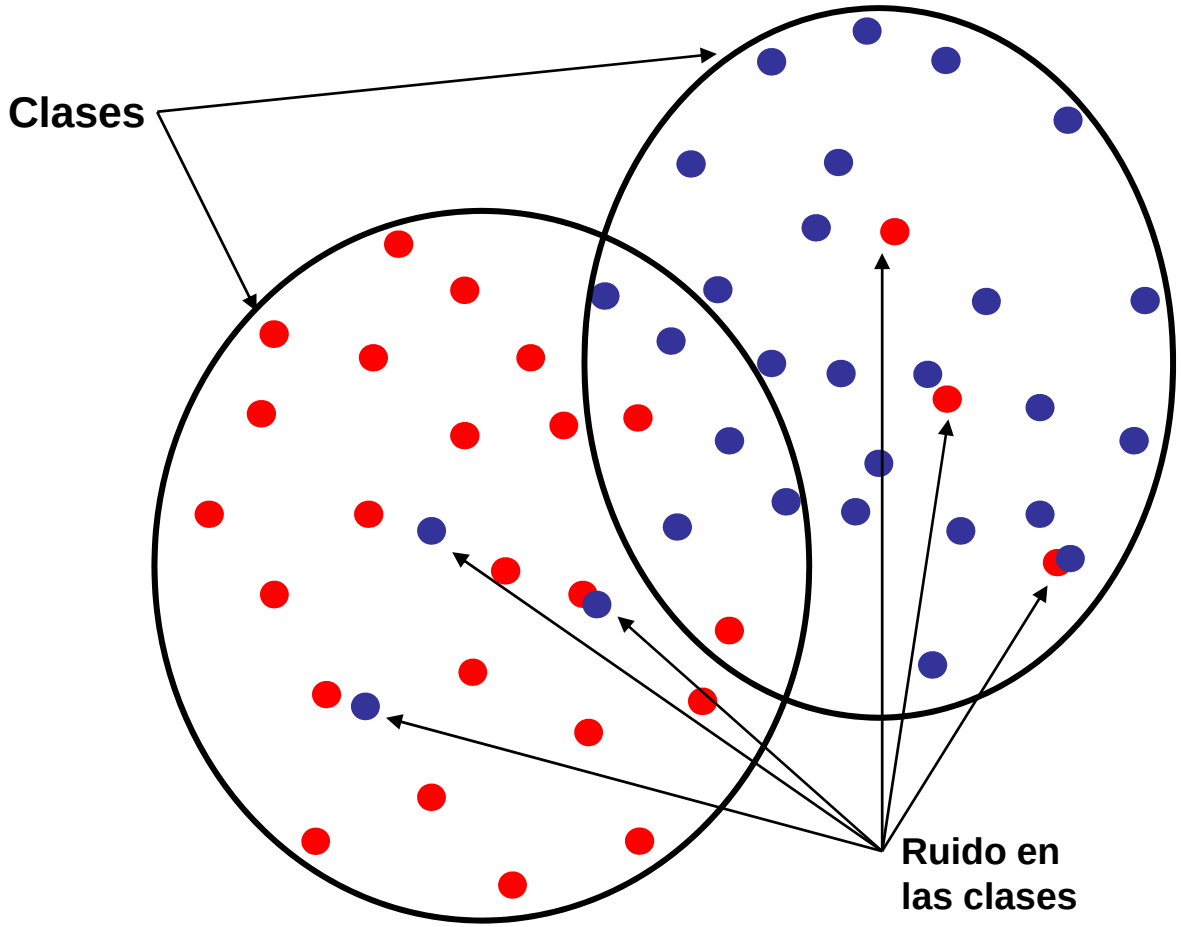
IV. Detección de ruido

- **Ruido en las variables**: Son los errores que se presentan al ingresar los valores de las variables. Las fuentes del ruido en las variables pueden ser:

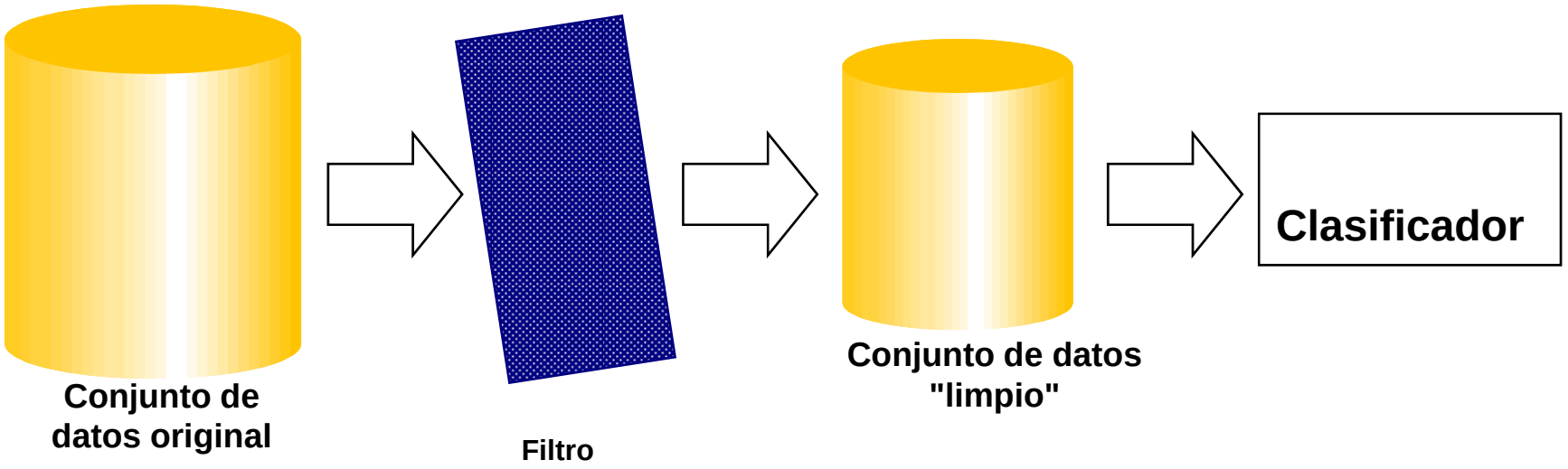
  - Errores en los valores de las variables.
  - Variables con valores perdidos o desconocidos.
  - Variables incompletas
  - Datos redundantes

- **Ruido en las clases**: Son errores introducidos al asignar las clases a las instancias. La presencia de ruido en las clases puede deberse a la subjetividad, errores al ingresar los datos o información errada para asignar una clase a la instancia.

Representación gráfica del ruido en las clases



Modelo para la identificación y eliminación de ruido en las clases



Detección de ruido usando filtros basados en clasificadores (Brodley y Friedl)

1. Elegir M algoritmos de clasificación.
2. El conjunto de entrenamiento E es dividido en N subconjuntos disjuntos (usando validación cruzada-N).
3. Con cada uno de los M algoritmos, se construyen los clasificadores utilizando N-1 subconjuntos.
4. Se utilizan los M algoritmos de clasificación para asignar las instancias a una clase, de esta forma, cada instancia recibe M etiquetas de pertenencia a una clase.
5. El filtro compara la clase original de cada instancia con las M asignaciones de los clasificadores y decide si debe o no debe remover la instancia usando diferentes criterios tales como el simple, por consenso o la decisión por mayoría.
6. El conjunto de entrenamiento filtrado se utiliza para construir el clasificador de nuevas instancias.

Algoritmo **QcleanNOISE** (Daza, 2007)

```
QcleanNOISE(Conjunto de Entrenamiento E) {

        For( cada instancia I_i en E)
                Hallar su medida de calidad Q(I_i)
        endFor

        CandNoise = Φ
        For( cada instancia I_i en E)
                if ( Q(I_i) < 0 )
                        CandNoise = CandNoise U { I_i }
        endFor

        For( cada instancia I_i en CandNOISE)
                Hallar sus k vecinos mas cercanos (NN) en E
                Count(I) =0
                For( cada NN de I_i )
                        if (Class(NN) ==Class(I_i) ) Count = Count +1
        endFor

        For( cada instancia I_i en CandNOISE)
                if (Count(I) < (k+1)/2 ) E = E - { I }
        endFor

        Return E
}
```

RESULTADOS

Para comparar el rendimiento se utiliza el error de clasificación y las siguientes medidas: ER1, ER2 y la precisión de la eliminación de ruido (NEP).

$$NEP = \frac{\|D \cap R\|}{\|D\|} \qquad ER_1 = \frac{\|D \cap \tilde{R}\|}{\|\tilde{R}\|} \qquad ER_2 = \frac{\|\tilde{D} \cap R\|}{\|R\|}$$

Donde:

D = Conjunto de instancias que han sido detectadas como ruido y se han eliminado.

R = Conjunto de instancias ruidosas.

$\tilde{D}$ y $\tilde{R}$ representan sus respectivos complementos.

Resultados de la eliminación de ruido en el conjunto de datos Iris

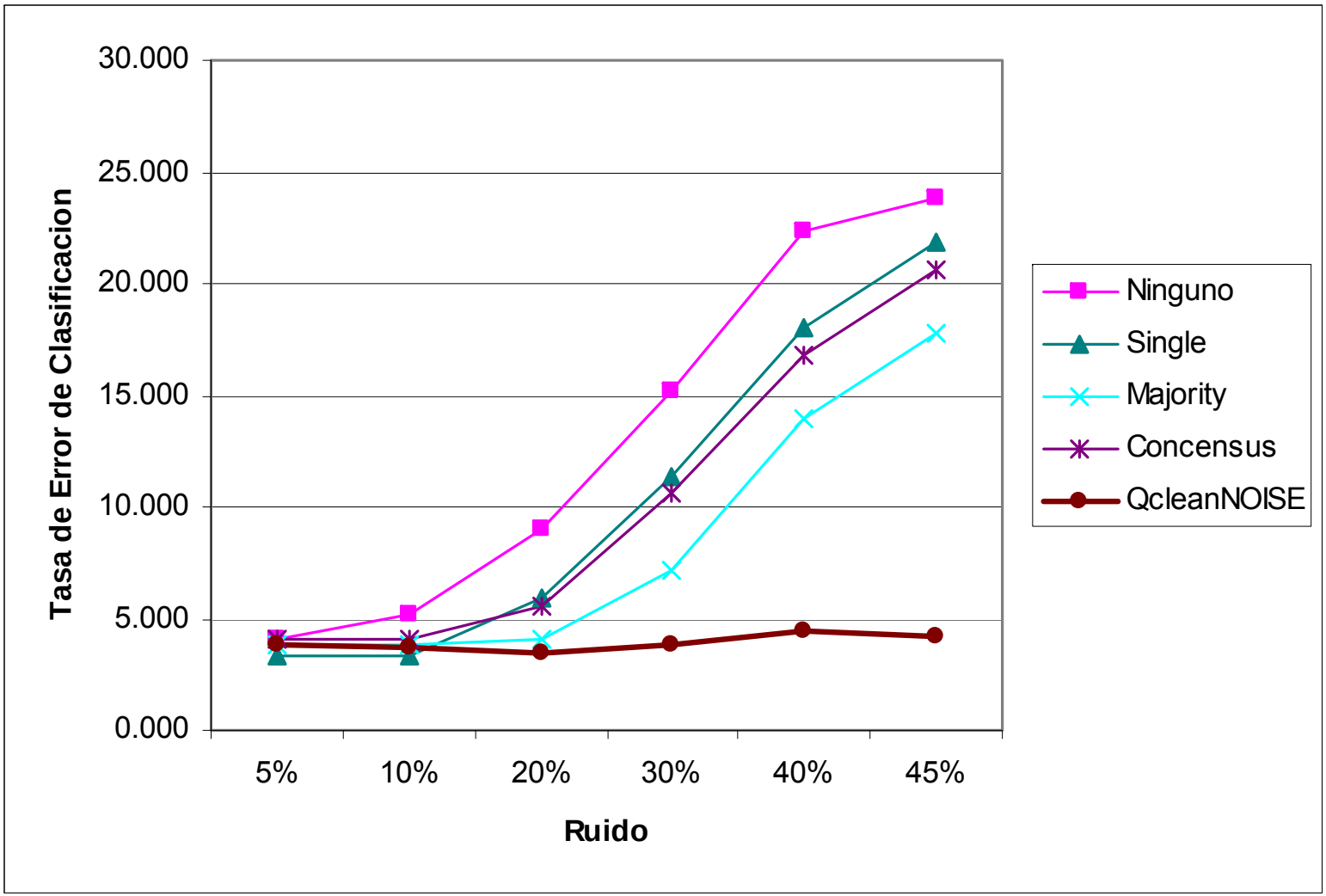
Ruido	QcleanNOISE			Single		
	ER1	ER2	NEP	ER1	ER2	NEP
5%	0.14	0.00	97.50	4.44	6.67	55.25
10%	0.22	2.22	97.88	10.12	12.67	49.98
20%	0.29	5.67	98.45	19.46	20.33	50.31
30%	0.37	16.52	98.56	27.75	32.22	50.38
40%	0.44	27.89	98.58	38.22	39.50	50.64
45%	0.59	41.09	98.35	42.68	46.07	49.98

Ruido	Majority			Consensus		
	ER1	ER2	NEP	ER1	ER2	NEP
5%	0.48	0.00	94.60	0.14	1.33	97.50
10%	0.48	0.00	96.67	0.19	5.33	98.10
20%	3.43	2.00	87.94	0.19	25.00	98.91
30%	11.72	10.44	75.94	1.46	41.56	94.55
40%	23.43	21.00	68.65	7.93	55.67	78.37
45%	37.84	35.56	57.31	10.94	66.67	70.22

Tasas de error de clasificación para los diferentes niveles de ruido y los métodos de detección en el conjunto de datos Iris usando KNN

		Brodley			
Ruido	**Ninguno**	**Single**	**Majority**	**Concensus**	**QcleanNOISE**
5%	4.133	3.289	3.778	4.133	3.822
10%	5.156	3.378	3.778	4.133	3.644
20%	9.067	5.911	4.133	5.511	3.511
30%	15.244	11.378	7.111	10.667	3.778
40%	22.400	18.000	14.000	16.844	4.444
45%	23.822	21.911	17.733	20.667	4.178

Comportamiento de las tasas de error de clasificación para los diferentes niveles de ruido y los métodos de detección en el conjunto de datos Iris usando KNN



V.Selección de Instancias

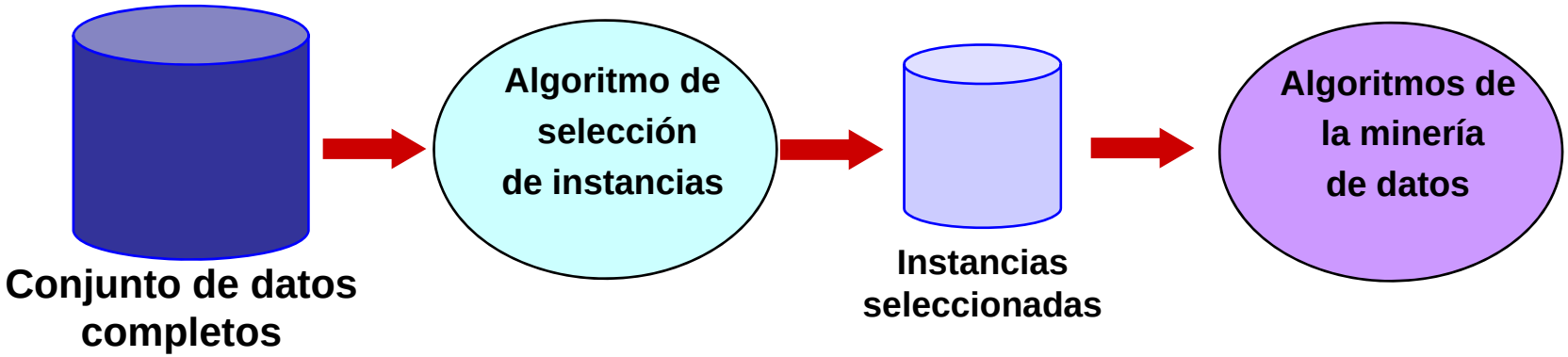
- **Selección de casos** (Liu y Motoda, 2002). Idealmente es un modelo independiente en el que dado un algoritmo A de minería de datos y su rendimiento R. Si ***s*** es la muestra de instancias seleccionadas y ***w*** el conjunto completo de datos, se debe tener:

$$R\left(A_s\right) \cong R\left(A_w\right)$$

- **Modelo independiente**. Dados dos algoritmos de DM, A^j y A^i , sea ΔR la diferencia en el rendimiento usando los datos **s** con respecto a usar los datos **w**,

$$\Delta R\left(A^i\right) \cong \Delta R\left(A^j\right)$$

Selección de Instancias



Problemas de la selección de casos

- Determinar el tamaño de muestra
- Garantizar la calidad de la muestra seleccionada
- Escalabilidad

Muestreo Progresivo

El muestreo progresivo, es un muestreo secuencial en el que inicialmente se selecciona una muestra pequeña y luego se incrementa progresivamente su tamaño hasta obtener un tamaño óptimo. Se tiene la secuencia de tamaños de muestra S

$$S = \{n_0, n_1, n_2, \cdots, n_k\} \qquad n_i < n_j$$

❖ **Muestreo progresivo aritmético** (John y Langley, 1996). La secuencia de muestras está definida por :

$$S = n_0 + i \times n_0 = \{n_0, n_0 + n_0, n_0 + 2n_0, \cdots, N\}$$

❖ **Muestreo progresivo geométrico** (Provost et al.,1999). El incremento del tamaño de la muestra se realiza a una razón geométrica. Con esta propuesta ellos buscan superar una de las limitaciones del muestreo progresivo aritmético. La secuencia de muestras, se define como:

$$S = a^i \times n_0 = \{a \times n_0, a^2 \times n_0, a^3 \times n_0, \cdots, N\}$$

Algoritmo CNN (Hart, 1968)

CNN (Conjunto de entrenamiento *E)*
S = Ø

Repetir

Add = FALSE
Para todas las Instancias en *E*, hacer:
Elegir aleatoriamente una instancia *I* de *E*
Hallar $I_c \in S$ tal que $dist(I, I_c) = \min_j dist(I, I_j)$
Si (clase (I) ≠ clase (I_c)) entonces
$S = S \cup \{I\}$
Add = TRUE

Parar Si (NOT (Add))

Retornar *S*

Algoritmo PCNN (Progresivo CNN, Daza 2007)

ProgCNN(Conjunto de Entrenamiento E, a, β) {

- sea i=0
- Sea S = Φ (subconjunto inicial vacío)
- n_0 = βN (tamaño de muestra inicial)

1. Seleccionar una muestra inicial de tamaño n_0
2. A la muestra de tamaño n_0 aplicar el algoritmo CNN
3. Las instancias elegidas en el paso 2 se agregan a S
4. Hallar Acc[0], teniendo a S como conjunto de entrenamiento.
5. While($n_0 a^{i+1}$ < N) do
6. i= i + 1
7. Seleccionar una muestra de tamaño $n_i = n_0(a^i - a^{i-1})$
8. A la muestra de tamaño n_i aplicar el algoritmo CNN
9. Las instancias elegidas en el paso 8 se agregan a S
10. Usando un algoritmo de clasificación, hallar Acc[i], teniendo a S como conjunto de entrenamiento.
11. Calcular ls1 <- acc[i-1]+sqrt(acc[i-1]*(1-acc[i-1])/n)
12. if((Acc[i] < ls1) & (Acc[i] > = Acc[i-1])) Return S
    else ir al paso 1.

Return S

Error de clasificación y tasa de reducción usando el clasificador RPART

Conjunto	Sin sel. Inst.		CNN		ProgCNN	
	Error	Reduc.	Error	Reduc.	Error	Reduc.
BUPA	31.68%	0%	40.14%	41	38.12%	74
SEGMENT	8.17%	0%	24.72%	87	10.91%	80
ABALONE	31.68%	0%	38.71%	41	38.87%	55
LANDSAT	18.66%	0%	21.04%	80	21.24%	83
WAVEFORM	26.63%	0%	26.68%	58	27.70%	69
PENBASED	18.21%	0%	27.39%	96	21.88%	93
SHUTTLE	0.53%	0%	0.65%	96	0.21%	99
Promedio	**19.37%**	**0.00%**	**25.62%**	**71**	**22.70%**	**79**

Tasa de reducción y costo computacional usando el clasificador RPART

Conjunto	CNN		ProgCNN	
	Reduc.	CCS	Reduc.	CCS
BUPA	41	60	74	26
SEGMENT	87	980	80	321
ABALONE	41	30251	55	2639
LANDSAT	80	109928	83	4951
WAVEFORM	58	332596	69	16780
PENBASED	96	10610	93	2135
SHUTTLE	96	7205	99	2465
Promedio	**71**	**70233**	**79**	**4188**

VII-CONCLUSIONES Y TRABAJO FUTURO

Conclusiones:

1. Las medidas propuestas para identificar el grado de complejidad de un problema de clasificación asociado a la estructura de los conjuntos de datos, son mas eficientes y precisas que las que están disponibles en la literatura. Además, ellas muestran altos niveles de correlación con los errores de clasificación.

2. El método filtro para seleccionar variables que se propone en esta investigación, mostró un mejor rendimiento que el método filtro ReliefF, en términos del error de clasificación y los tiempos de ejecución.

3. La presencia de ruido en las clases produce un deterioro en el rendimiento de los clasificadores. Sin embargo, el grado del efecto de la presencia de ruido varía de un clasificador a otro. Por ejemplo, el clasificador basado en los vecinos más cercanos (KNN), es más sensible a la presencia de ruido en las clases que los clasificadores LDA y RPART.

CONCLUSIONES Y TRABAJO FUTURO

Conclusiones:

4. La detección y posterior eliminación del ruido en las clases en el conjunto de entrenamiento de un problema de clasificación supervisada, es un paso importante en el pre-procesamiento de los datos ya que produce una mejora en el nivel de precisión de los clasificadores que hacen uso del conjunto de entrenamiento que ha sido "limpiado".

5. El algoritmo propuesto para la detección de ruido en las clases en el conjunto de entrenamiento (QcleanNOISE), muestra mejores resultados a altos niveles de ruido que los métodos para filtrar el ruido basados en clasificadores . Esto se debe a que los clasificadores son sensibles a la presencia de ruido y por ende tienden a perder precisión a altos niveles de ruido.

CONCLUSIONES Y TRABAJO FUTURO

Conclusiones:

6. El método para detectar ruido QcleanNOISE, presenta una mejor eficiencia computacional que los algoritmos basados en clasificadores.

7. El muestreo progresivo permite mejorar la escalabilidad de los algoritmos de selección de instancias.

8. El uso del muestreo progresivo permite mejorar el rendimiento de los algoritmos de selección de instancias al incrementar la tasa de reducción con menores costos computacionales.

CONCLUSIONES Y TRABAJO FUTURO

Trabajo futuro:

1. Extender las medidas de complejidad propuestas a conjuntos de datos con valores perdidos.

2. Aplicar las medidas de complejidad propuestas a la selección de instancias, para elaborar algoritmos eficientes que reduzcan el conjunto de entrenamiento en problemas de clasificación supervisada.

3. Aplicar los métodos de selección de variables a conjuntos de datos provenientes de experimentos de microarreglos, en los cuales se tiene una gran cantidad de variables predictoras.

4. Analizar la aplicación y adaptación de las medidas de complejidad a problemas de detección de *outliers* y de clasificación no supervisada.

5. Aplicar métodos ponderados de selección de muestras, usando estimaciones no paramétricas para los pesos de las instancias.

An algorithm for detecting noise on supervised classification

Edgar Acuña and Luis Daza

Mathematical Science Department
University of Puerto Rico at Mayaguez

Motivation

- The ideal situation for extracting knowledge from a dataset is when it does has neither outliers nor noise. However, real-world databases are usually dirty and a cleaning process is required before performing a data mining task.
- The noise in a dataset may deteriorate the performance of a classifier applied on it, since the misclassification error as well as the computing time may increase and the classifier and decision rules obtained could be more complex.
- In a supervised classification context, the quality of a dataset is characterized by two information sources: the predictor attributes and the categorical attribute which defines the classes. The quality of the predictors is determined by their quality to represent the instances to be classified, and the quality of the class attribute is determined by the correct assignment of each instance.

Types of Noise

a) **Noise in the attributes**: It is given by the errors occurred during the entrance of the values of the attributes. Among the sources of this type of noise are: variables with missing values, and redundant data.

b) **Noise in the classes**: It is given by the errors introduced during the assignment of the instances to the classes. The presence of this kind of noise may be due to subjectivity, errors in the data entry process, and incorrect information for assigning a instance to a class. There are two possible sources of class noise: i) Inconsistent instances, and ii) Instances assigned incorrectly to a class.

- There are several procedures to identify noisy instances. Guyon *et al.* (1996) use the criterion of the maximum information gain, Gamberger, *et al*., (1999), use a method called *saturation filter*.
- In other methods, the instances considered potentially as noise are detected and removed using C4.5 (John 1995, Zhu *et al*., 2003), or neural networks (Zeng and Martinez, 2003). .
- Brodley and Friedl (1996, 1999), propose a model for filtering instances. The initial training set that contains noise is filtered using several strategies in order to obtain a new clean dataset, which is used to construct the classifiers.. There are several variants of the method depending on the criterion used to consider an instance as noise as well on the number of classifiers used as filter. In case that only one classifier is used as filter, every instance incorrectly classified is considered as noise and it must be removed. For several classifiers either majority voting or consensus voting is used.
- Zhu *et al*., (2003, 2006), focused in the problem of cleaning large and distributed databases.

A data quality measure

- To evaluate the quality of the i-th instance, we compute $Q_i=(r_i-d_i)/max(d_i,r_i)$,

where d_i is the distance of the i-th instance to the centroid of its class and r_i is the minimum distance of the i-th instance to the centroid of the classes where does not belong to. A noisy instance will have negative values for the quality measure Q. However, some instances located near to the boundary of two or more classes may also have small negative values for the quality measure.

The QcleanNoise Algorithm

- Our noise detection algorithm detects first the potential noisy instances where Q<0.
- Then for each of these noisy instances the behavior of its k neighbors is checked. If most of the neighbors belongs to a different class of the given instance, then this is considered as noisy and it is discarded from the training set.

The QcleanNoise algorithm

```
QcleanNOISE(Training Dataset E) {
    For( each instance Ii in E)
        Find out its quality measure Q(Ii)
    endFor
      CandNoise = Φ
    For( each instancei in E)
        if ( Q(Ii) < 0 )
                CandNoise = CandNoise U { Ii }
    endFor
    For( each instance i in CandNOISE)
        Find out its  k nearest neighbors NN in E
        Count(I) =0
        For( each NN of i )
                if (Class(NN) ==Class(i) ) Count = Count +1
    endFor
    For( each instance i in CandNOISE)
        if (Count(I) < (k+1)/2 ) E = E - { I }
    endFor
    Return E containing only non-noisy instances
}
```

Methodology

- First, we insert randomly noise in the class labels of the dataset at several percentages: 5,10,20,30,40,45.
- The proposed technique identifies the noisy instances, which then are removed from the training data. The remaining instances are used to construct the classifier and the misclassification error rate is estimated using cross-validation.
- This noise removal process is applied to four datasets; Iris, Breastw, Segment, Landsat; using three classifiers: Linear Discriminant Analysis (LDA) , a decision tree classifier based on recursive partitioning (RPART) and the k-nn classifier (KNN).

Measures to evaluate the effectiveness of the noise detection methods

- Brodley and Friedl, 1999, and Zhu *et al.*, 2003;

$$NEP = \frac{\|D \cap R\|}{\|D\|} \qquad ER_1 = \frac{\|D \cap \tilde{R}\|}{\|\tilde{R}\|} \qquad ER_2 = \frac{\|\tilde{D} \cap R\|}{\|R\|}$$

where, D = set of instances that are detected as noise and are eliminated, R = set of noisy instances. and $\tilde{D}$ and $\tilde{R}$ represent their respective complement. ER_1, occurs when a non-noisy instance is considered as noise, ER_2, occurs when a noisy instance is considered as an instance correctly labeled. The *NEP* measures the proportion of noisy instances that are detected as noise.

Table I. Measures of noise removal on the Iris dataset

Noise	QcleanNOISE			Single			Majority			Consensus		
	ER1	ER2	NEP	ER1	ER2	NEP	ER1	ER2	NEP	ER1	ER2	NEP
5%	0.1	0.0	97.5	4.4	6.7	55.3	0.5	0.0	94.6	0.1	1.3	97.5
10%	0.2	2.2	97.9	10.1	12.7	50.0	0.5	0.0	96.7	0.2	5.3	98.1
20%	0.3	5.7	98.5	19.5	20.3	50.3	3.4	2.0	87.9	0.2	25.0	98.9
30%	0.4	16.5	98.6	27.7	32.2	50.4	11.7	10.4	75.9	1.5	41.6	94.5
40%	0.4	27.9	98.6	38.2	39.5	50.6	23.4	21.0	68.7	7.9	55.7	78.4
45%	0.6	41.1	98.3	42.7	46.1	50.0	37.8	35.6	57.3	10.9	66.7	70.2

Table II. Misclassification error rates for the LDA classifier at different levels of noise removal using four detection methods on the Iris dataset.

		Brodley			
Noise	**None**	**Single**	**Majority**	**Consensus**	**QcleanNOISE**
5%	4.7	2.0	2.6	2.1	2.0
10%	7.0	2.2	2.5	2.3	2.1
20%	10.7	3.7	3.0	4.3	2.1
30%	13.2	8.6	5.3	9.1	2.2
40%	15.5	14.2	11.2	13.5	2.6
45%	22.7	20.4	16.9	19.6	2.3

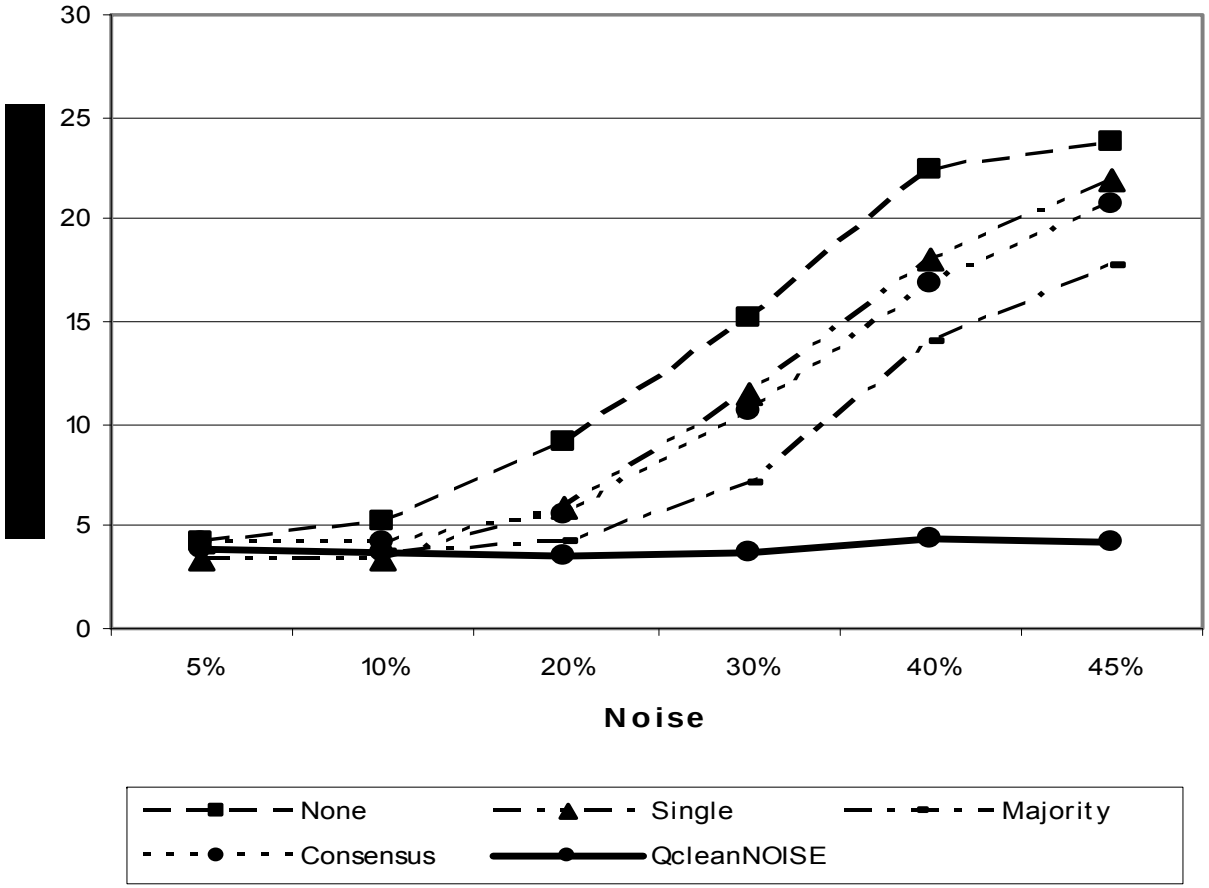


Figure 3. Misclassification error rates for the KNN classifier at different levels of noise removal using four detection methods on the Iris dataset

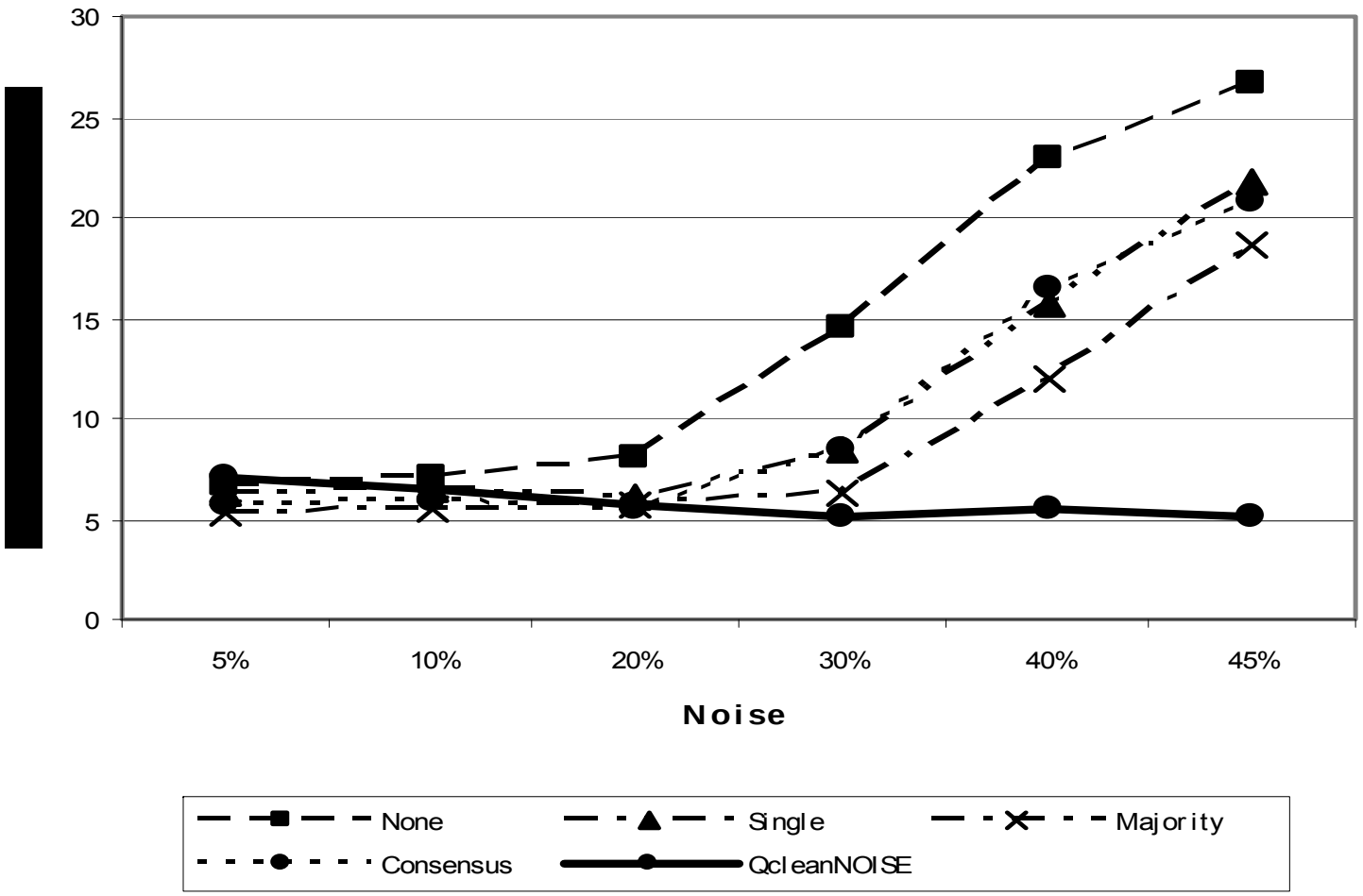


Figure 4. Misclassification error rates for the RPART classifier at different levels of noise removal using four detection methods on the Iris dataset

Table III. Measures of noise renoval on the Segment dataset

Noise	QcleanNOISE			Single			Majority			Consensus		
	ER1	ER2	NEP	ER1	ER2	NEP	ER1	ER2	NEP	ER1	ER2	NEP
5%	0.2	0.1	95.2	5.0	2.8	48.5	0.4	0.0	94.4	0.1	5.7	96.7
10%	0.2	0.8	96.3	9.6	8.7	49.3	0.4	0.1	94.8	0.1	8.2	97.7
20%	0.3	6.8	97.3	18.9	20.1	49.6	0.7	0.2	95.7	0.1	19.5	98.6
30%	0.4	19.2	97.3	28.0	30.1	49.8	3.5	2.9	90.5	0.3	34.4	98.1
40%	0.4	33.2	97.1	38.0	38.8	49.8	15.1	13.9	77.7	2.2	52.1	93.0
45%	0.5	42.0	96.9	42.3	44.8	49.7	29.2	29.5	64.9	6.3	64.5	81.0

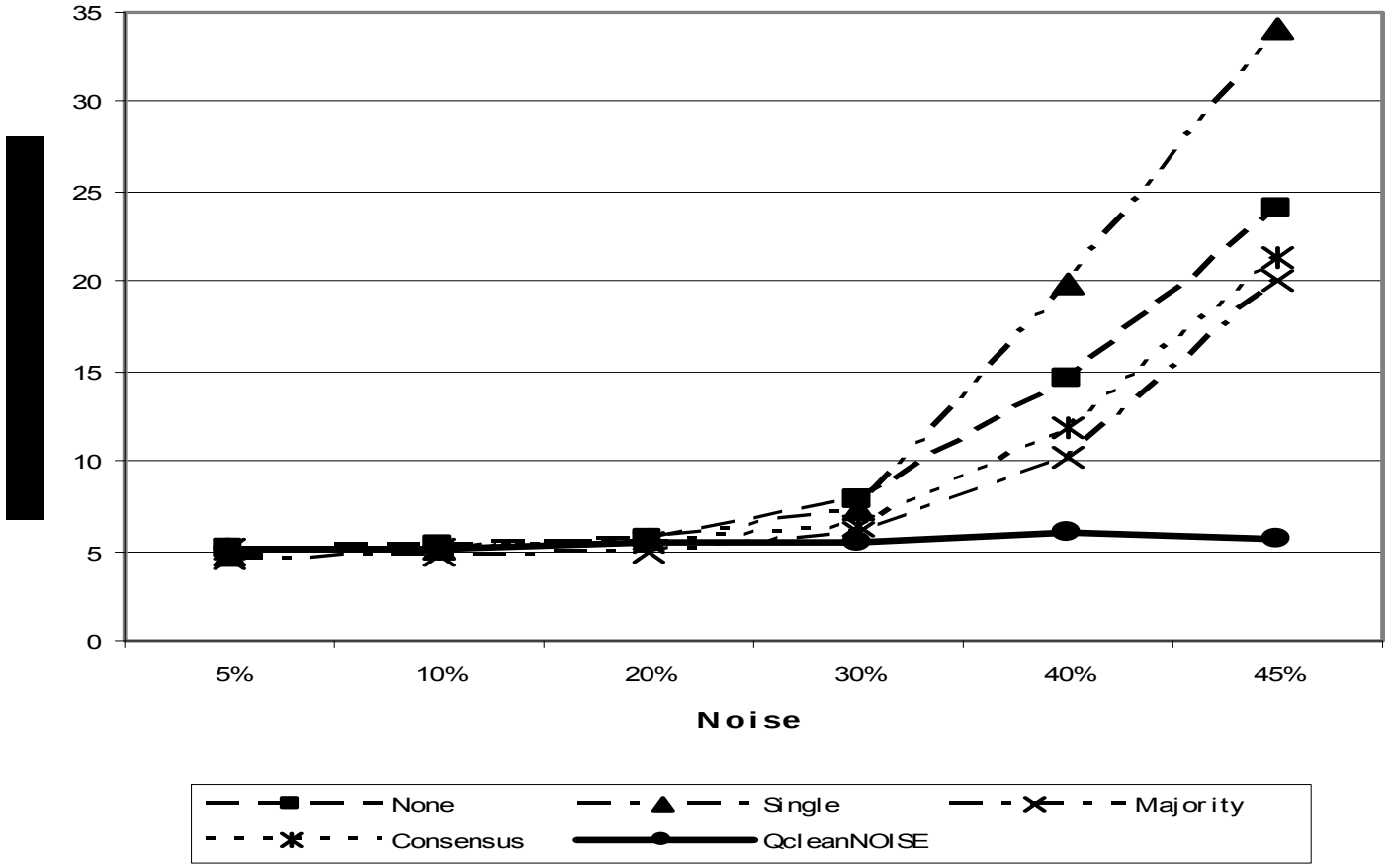


Figure 5. Misclassification error rates for the RPART classifier at different levels of noise removal using four detection methods on the Breastw dataset

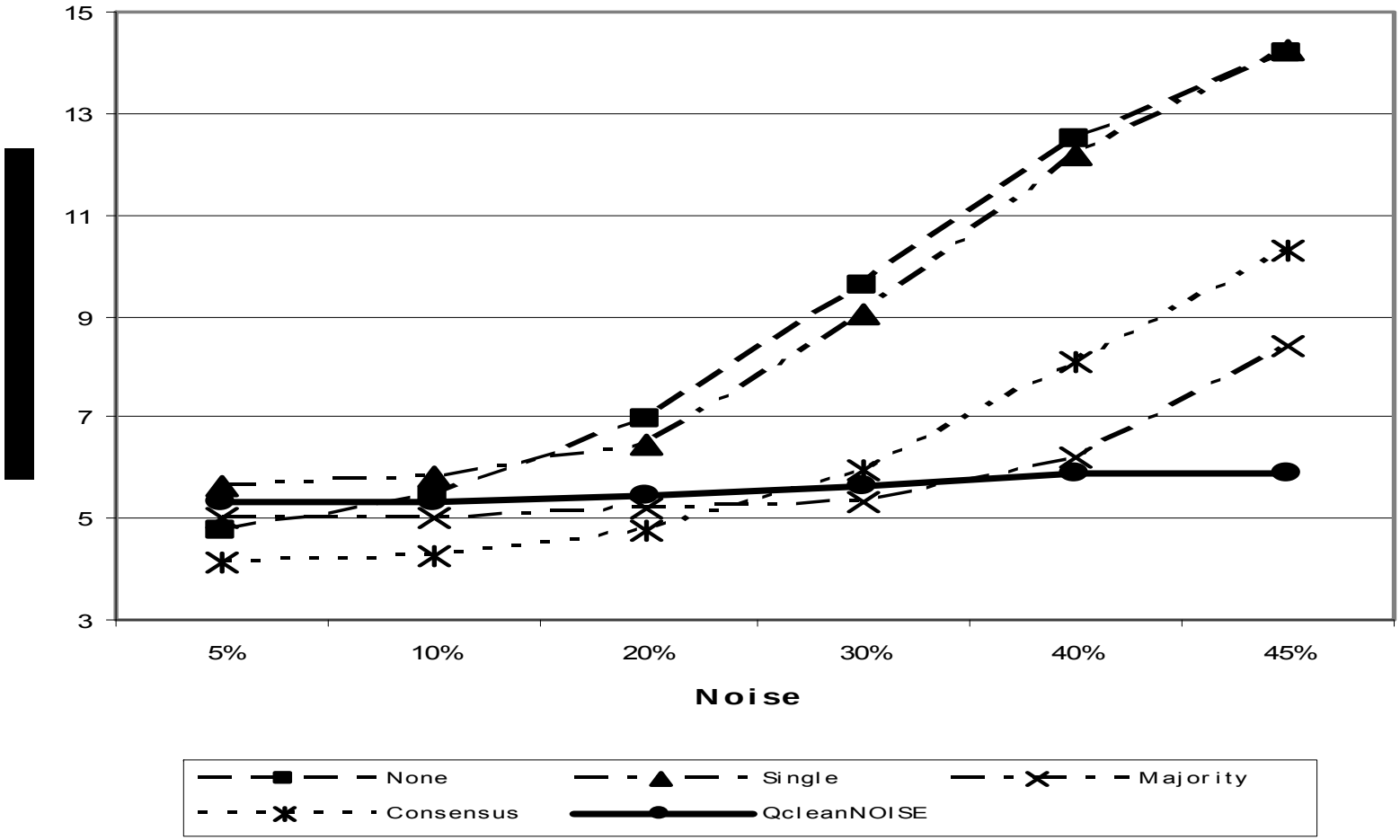


Figure 6. Misclassification error rates for the KNN classifier at different levels of noise removal using four detection methods on the Segment data

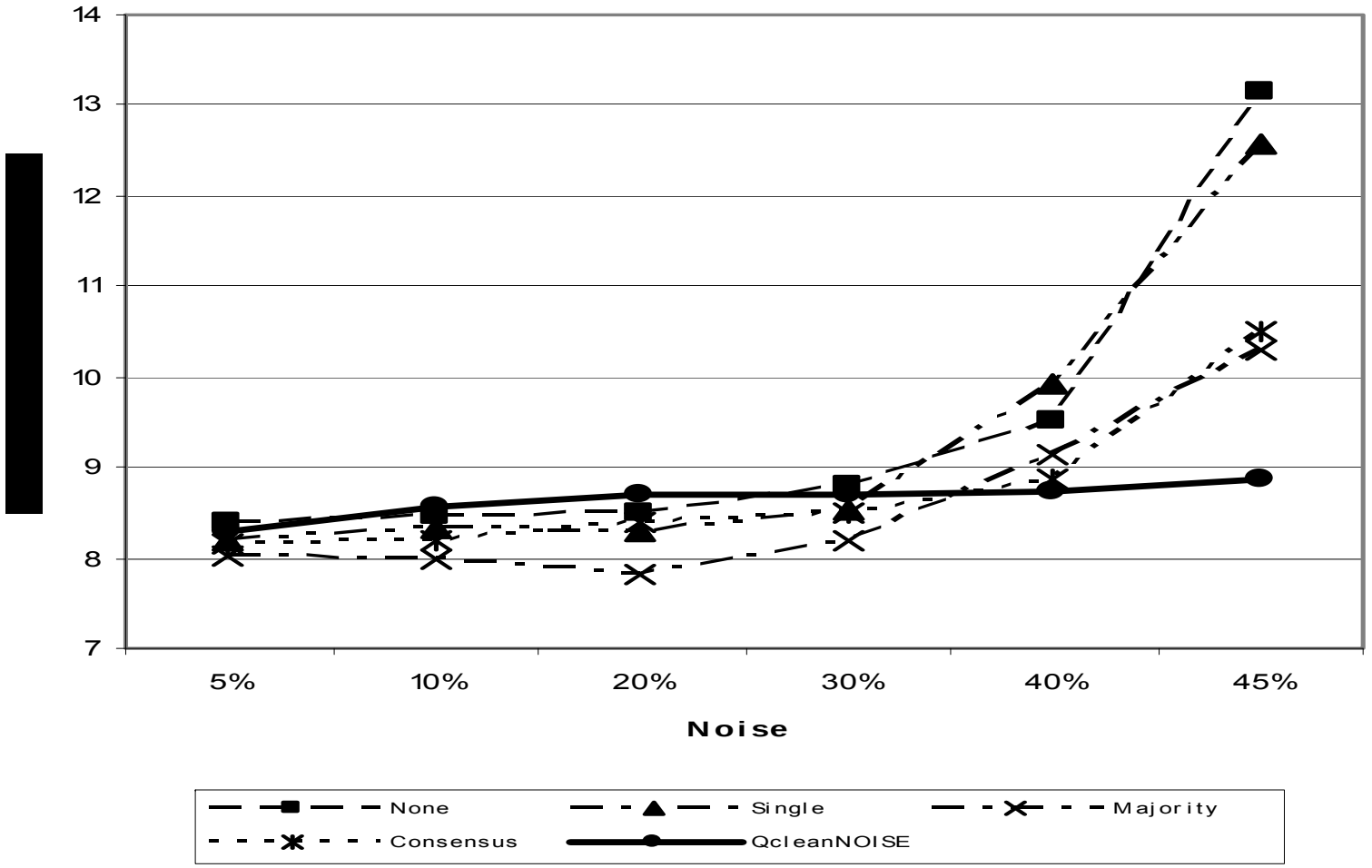


Figure 7. Misclassification error rates for the RPART classifier at different levels of noise removal using four detection methods on the Segment dataset

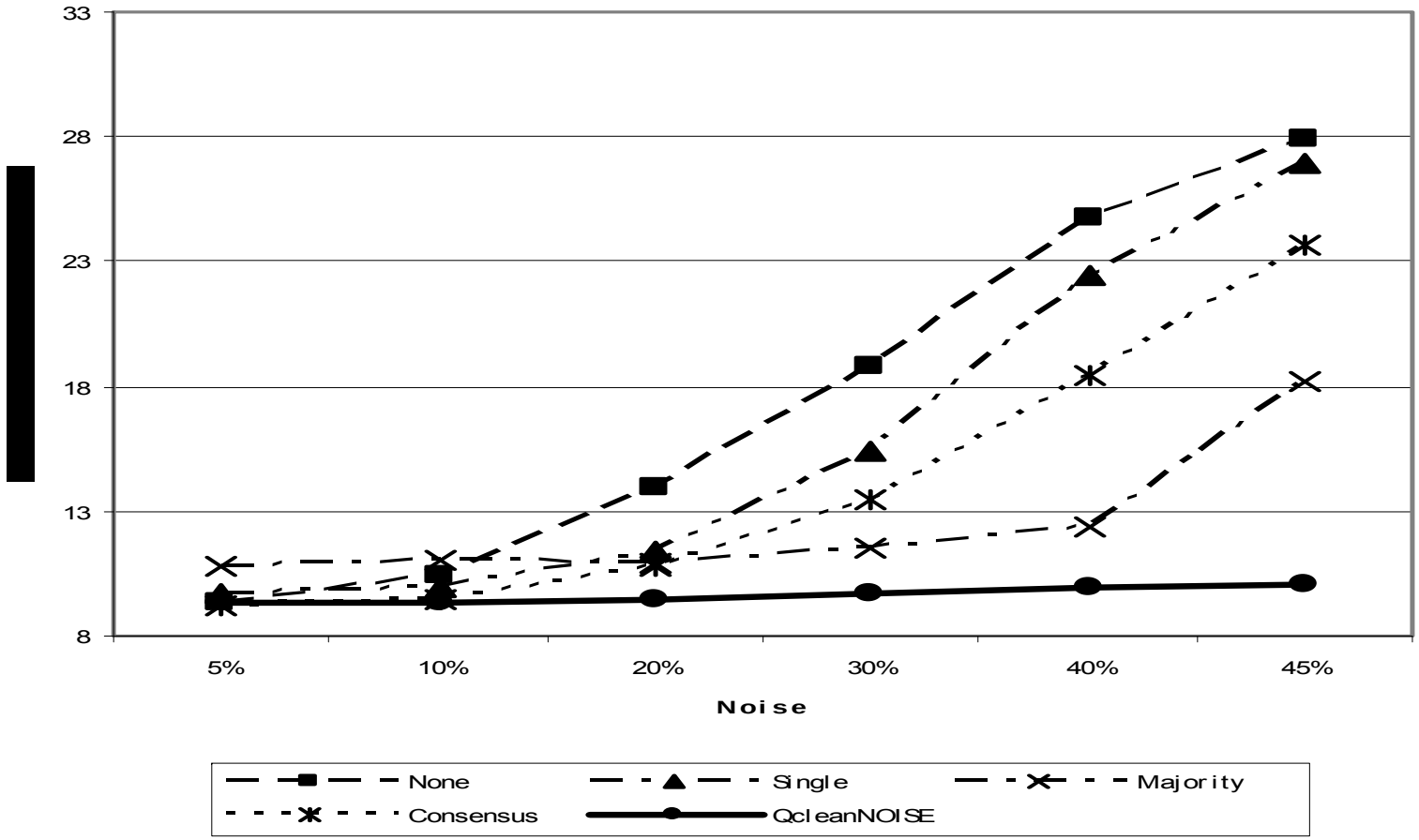


Figure 8. Misclassification error rates for the KNN classifier at different levels of noise removal using four detection methods on the Landsat dataset

Conclusion

- **Our empirical results show that QcleanNoise, which it is based on a data quality measure, outperforms strategies proposed by Brodley and Friedl for detecting noise.**
- **Our algorithm gives better results for the measures used to evaluate noise removal methods.**
- **Also, classifiers yield lower misclassification error rates after that the training dataset is cleaned out using QcleanNoise.**
- **KNN classifier seems to be the one most affected for the presence of noise.**