Universidad Nacional Autónoma de México.

Facultad de Estudios Superiores Acatlán.

DETERMINACIÓN DEL TAMAÑO MUESTRAL ÓPTIMO BASADO EN ENTROPÍA Y OPTIMIZACIÓN HEURÍSTICA.

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN MATEMÁTICAS APLICADAS Y COMPUTACIÓN.

PRESENTA:

JUAN CARLOS ALFARO PÉREZ.

ASESOR:
DR. EDWYN JAVIER ALDANA BOBADILLA

OCTUBRE, 2016.

Santa Cruz Acatlán, Naucalpan, Edo. de Méx.

A mis padres, Humberto y Esther. Les agradezco profundamente su amor expresado en las numerosas formas que me han acompañado a lo largo de mi vida y que han contribuido a mi crecimiento.

A Kari, porque el sueño que te vencía como consecuencia de acompañarme en las madrugadas muchas veces fue el catalizador para seguir adelante con más entusiasmo.

A mi asesor, por sus clases que motivaron la búsqueda de su orientación en este trabajo, y por haberme permitido explorar con usted este espacio que trajo mucho aprendizaje y descubrimientos.

A Erika, porque desde pequeños, en una charla de esas en que deshojábamos un poco al buen Benjamín, anticipaste que llegaría este momento y has confiado en mí desde entonces.

A Hans, porque nos permitimos reinventar y fortalecer el lazo de amistad que ahora nos une.

.

A mis amigos más distantes en el tiempo: Luis, Aoi, Sergio, Kry, Alex, Reyna, Alina.

A los más recientes: Jordi, Abhi, Amador.

A Armando Reyes, por compartir tu entendimiento sobre Cortázar.

A Armando, por enseñarme que la amistad es una oportunidad de invención, por todo lo que aprendo de ti.

A todos los que han sido y a todos los que son parte esencial de mi vida.

"Tenemos que obligar a la realidad a que responda a nuestros sueños, hay que seguir soñando hasta abolir la falsa frontera entre lo ilusorio y lo tangible, hasta realizarnos y descubrir que el paraíso está ahí, a la vuelta de todas las esquinas".

-Julio Cortázar, Entrevista Alcor 29, 1964-

Resumen.

Una tarea común en el análisis de datos es encontrar una muestra adecuada cuyas propiedades nos permitan inferir los parámetros y el comportamiento de la población. Un problema usual al respecto es determinar el tamaño óptimo de la muestra. Para resorverlo, la estadística provee algunas técnicas usualmente basadas en los resultados asintóticos del Teorema del Límite Central. Sin embargo, la eficiencia de tales métodos está limitada por diversas consideraciones como la estrategia de muestreo (simple: con o sin reemplazo, estratificado, basado en clusters, etc.), el tamaño de la población y la dimensionalidad del espacio de los datos. A fin de evitar estas limitaciones, se sugiere un método basado en la información del conjunto de datos en términos de la Entropía de Shannon.

Se propone encontrar una muestra óptima de tamaño n cuya cantidad de información sea lo más cercana posible a la cantidad de información de la población P. Existen muchas formas de seleccionar una muestra de tamaño n de la población, que satisfaga lo anterior. Esto plantea un problema combinatorio con un espacio de búsqueda que deshabilita el uso de técnicas tradicionales. Por esta razón se aplicó una búsqueda heurística basada en un algoritmo genético. Con base en un estudio preliminar se decidió usar un algoritmo genético denominado Algoritmo Genético Ecléctico (EGA) el cual ha demostrado resolver eficientemente un amplio acervo de problemas. Planteado el problema de muestreo como un problema de búsqueda heurística y establecido el método para resolverlo, se realizó un conjunto de experimentos utilizando datos sintéticos y datos reales obtenidos de diferentes contextos de aplicación. Dichos experimentos mostraron una gran efectividad del método propuesto, permitiendo además inferir múltiples escenarios de aplicación.

Abstract.

A common task in data analysis is to find the appropriate data sample whose properties allow us to infer the parameters of the data population. The most frequently dilemma related to sampling is how to determine the optimal size of the sample. To solve it, statistics offers some sample techniques usually based on asymptotical results from the Central Limit Theorem. However, the effectiveness of such methods is bounded by several considerations as sampling strategy (simple: with or without replacement, stratified, cluster-based, etc.), the size of the population and the dimensionality of the space of the data. To avoid such constraints, we propose a method based on a measure of information of the data in terms of Shannon's Entropy.

In this sense, the intention is to find an optimal sample of size n whose information is as similar as possible to the information of the population P, subject to several constraints. Finding such sample represents a hard optimization problem whose feasible space disallows the use of traditional optimization techniques, so it became necessary the use of a heuristic method. Based on a preliminary study it was determined to use an algorithm called Eclectic Genetic Algorithm (EGA) which showed to solve a wide collection of problems. The method was evaluated using synthetic datasets; the results showed that is suitable. For completeness, it was used a dataset from a real problem; the results confirm the effectiveness of the proposal and allow us to visualize different applications.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción.

La minería de datos (MD) es una disciplina que involucra el análisis y procesamiento de *datos* con el fin de inferir *información* respecto a determinado fenómeno. Esto implica el uso de diversas técnicas basadas habitualmente en la estadística y la computación. Es de vital importancia que los datos involucrados durante los procesos de MD sean consistentes y representativos del fenómeno mencionado.

Para asegurar esto, la MD requiere de procesos preliminares conocidos como *cleaning* (remoción de ruido, inconsistencias, datos redundantes) e *integration* (consolidación de diferentes fuentes de datos en repositorios unificados, continuamente denominados *data-warehouses* [1]) .

El proceso de cleaning permite resolver problemas como datos incompletos [2][3][4], datos altamente correlacionados [5], alta dimensionalidad [6], datos categóricos [7][8] y conjuntos de datos de alta cardinalidad.

La *complejidad* computacional de todo proceso de MD está en función tanto de la cardinalidad como de la dimensionalidad del conjunto de datos (de ahora en adelante P), lo cual implica en muchos casos, complejidades en tiempo y espacio no polinomiales [9][10]. Respecto a la cardinalidad de P (denotada como N), lo más común es recurrir a técnicas estadísticas de muestreo, que permiten reducir el conjunto de datos (*población*) a un subconjunto denominado *muestra.* Uno de los principales inconvenientes es determinar el *tamaño óptimo* de dicha muestra. En algunos casos, cuando P es pequeño (típicamente entre 5 y 30 elementos) se recurre al uso de toda la población, lo cual permite inferir información con un alto nivel de significancia [11]. Esta no es una alternativa viable cuando el tamaño de la población es del orden $1X10^k$ para $k \geq 3$. En

las siguientes secciones se presentan algunas estrategias comunes para encontrar una muestra representativa S y se discute acerca del problema de determinar su cardinalidad.

1.1. Seleccionando los elementos de la población.

Un método tradicional de muestreo es el *muestreo aleatorio simple* [12] en donde cada elemento de la población tiene la misma probabilidad de ser incluido en la muestra. Si se decide considerar nuevamente un elemento ya seleccionado de la población, se habla de *muestreo aleatorio con reemplazo*, en caso contrario, hablamos de un *muestreo aleatorio sin reemplazo*. Otros métodos frecuentemente empleados son: el *muestreo sistemático*, en donde el tamaño de la población es dividido entre el tamaño deseado de la muestra, con el fin de obtener la amplitud de los intervalos de muestreo [13], el *muestreo estratificado*, en el cual la población es repartida en grupos o estratos de datos no sobrepuestos; una muestra es obtenida de cada estrato aplicando muestreo simple [14]. El *muestreo de clusters* [15], al igual que el muestreo estratificado, distribuye a la población en clusters o grupos con la diferencia de que la unidad de muestreo (elemento de la muestra) es el cluster y no los elementos dentro de él.

Suponiendo que todas las posibles muestras de tamaño n son obtenidas sin reemplazo desde una población finita de tamaño $N > n$, la media y la desviación estándar de la distribución de muestreo de medias denotadas por $\mu_{\bar{X}}$ y $\sigma_{\bar{X}}$ están dadas por:

$$\mu_{\bar{X}} = \mu \qquad \sigma = \frac{\sigma}{\sqrt{n}}\sqrt{\frac{N-n}{N-1}} \tag{1.1}$$

Donde μ y σ son la media y la desviación estándar de P, respectivamente [16]. Por otro lado, si la población es infinita o la forma de seleccionar las muestras es con reemplazo, la ecuación anterior se reduce a:

$$\mu_{\bar{X}} = \mu \qquad \sigma_{\bar{X}} = \frac{\sigma}{\sqrt{n}} \tag{1.2}$$

Para valores grandes de $n(n \geq 30)$ la distribución de muestreo de medias es aproximadamente normal como consecuencia del Teorema del Límite Central (TLC) [17]. Con base en este hecho es posible realizar inferencias acerca de P considerando la normalidad de la distribución muestral de sus medias. Claramente la significancia estadística de dichas inferencias dependerá del valor de n y del número de muestras M que hacen que la distribución de las mismas sea aproximadamente normal.

Dado que $\sigma_{\bar{x}}$ representa una medida de dispersión de las muestras (tradicionalmente llamado error estándar), su valor tiene que ser lo más pequeño posible. Existe un valor óptimo de n que permite satisfacer tal condición. En la figura 1.1, se ilustra este hecho con un conjunto de datos $\mathbb{R}$, de cardinalidad 6000. Cada punto es el error estándar obtenido con diferentes valores de n, asumiendo que la estrategia de muestreo es con reemplazo. Se puede observar que el valor de n que minimiza el error estándar es el más cercano al tamaño de la población. Evidentemente, para propósitos prácticos, este valor es inapropiado. Un valor de n ideal será aquel que sea menor a la cardinalidad de la población.

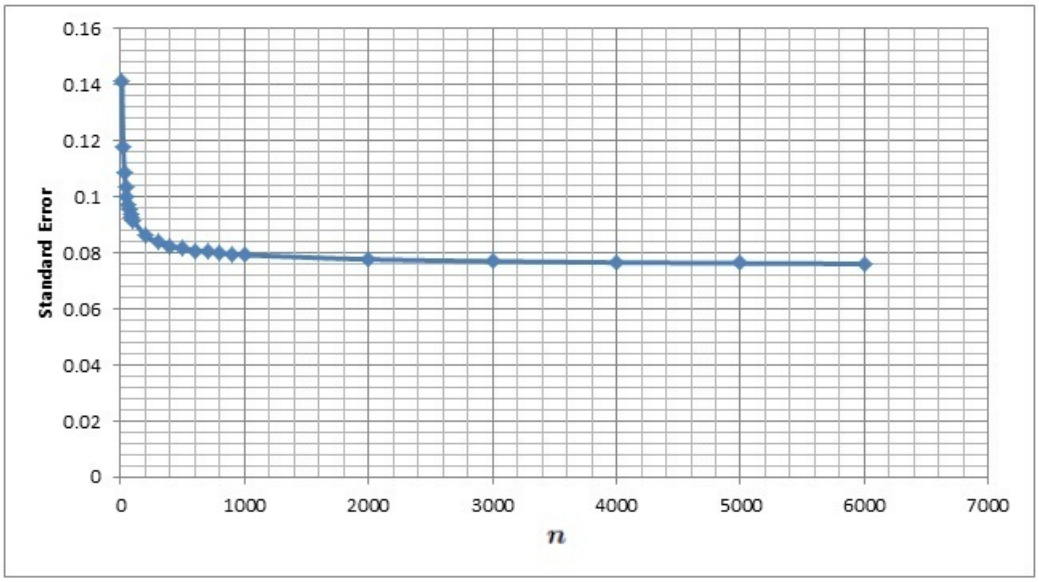

Figura 1.1: Error estándar en función del tamaño de la muestra.

1.2. Determinando el valor de n.

Con base en (1.2) el tamaño de la muestra (con reemplazo) está dado por:

$$n \cong \frac{\sigma^2}{\sigma_{\bar{x}}^2} \tag{1.3}$$

Típicamente el valor de $\sigma_{\bar{x}}$ se define de manera discrecional y representa el error permisible en el proceso de muestreo. Retomando el ejemplo mostrado en la figura 1.1, si se asume que $\sigma_{\bar{x}} = 0.05$, el valor de n será mucho mayor al tamaño o cardinalidad de P. Esto significa que no siempre se obtiene un valor de n adecuado, dado un valor de $\sigma_{\bar{x}}$. Es necesario encontrar el punto de equilibrio entre n y $\sigma_{\bar{x}}$. Una observación interesante es que generalmente las relaciones asintóticas en (1.2) asumen que las muestras son obtenidas a través de técnicas de muestreo aleatorio simple, usualmente métodos de muestreo más complejos no son tomados en cuenta. Otra consideración importante es que comúnmente las relaciones asintóticas determinadas por el TLC no consideran aquellas sobre los datos que pertenecen a espacios multivariados. Con base en lo anterior, en este trabajo se propone un método que permite encontrar la muestra óptima (aquella cuyo tamaño sea lo más pequeño posible y conserve la *información* de la población) sin recurrir a resultados asintóticos y que puede ser aplicado a datos en espacios multivariados ($\Re^n$).

En los capítulos y secciones siguientes se presentarán las ideas fundamentales del método y los experimentos que permiten corroborar su eficiencia. Este trabajo se ha organizado de la siguiente manera:

- En el Capítulo 2 se describen algunos trabajos relacionados al problema planteado y se propone una solución al mismo; se muestra cómo medir la información de un conjunto de datos con base a la Entropía de Shannon y de qué manera extender este concepto a un espacio multidimensional.

- El Capítulo 3 plantea el problema de encontrar la muestra adecuada, como un problema de optimización.

- En el Capítulo 4 se presenta la forma de resolver el problema a través de una técnica heurística, en particular un Algoritmo Genético.

- El capítulo 5, muestra la metodología experimental y sus resultados. Finalmente, se presentan algunas conclusiones y se infieren múltiples aplicaciones.

Es importante resaltar que como resultado de este trabajo se publicó un artículo, el cual está disponible en el Apéndice D: Paper, para el lector interesado.

Capítulo 2

Planteamiento del problema.

El problema de encontrar una muestra apropiada se puede abordar desde dos escenarios:

1. Los parámetros y el comportamiento de P son desconocidos y estos se deben inferir a partir de un limitado conjunto de observaciones S.

2. Existe un extenso conjunto de observaciones P, pero debido a restricciones (de cómputo, geográficas, económicas, etc.) se requiere obtener un conjunto S cuyos parámetros y comportamiento sean tan similares como los de P.

El primer escenario es común en contextos en los que determinar los parámetros de P puede ser poco práctico o no factible. Por ejemplo, estudios médicos en donde los datos involucran pacientes con cierta enfermedad o encuestas sobre las preferencias políticas de los habitantes de alguna ciudad o país, en los cuales disponer de la totalidad de la población es muy costoso o definitivamente inviable. En estos casos, generalmente el muestreo se realiza a través de los resultados asintóticos del TLC. Como los parámetros de P no se conocen, durante el muestreo se pueden asumir ciertas características sobre ellos, lo cual podría representar una desventaja. Debido a esto han surgido otras estrategias de muestreo, por ejemplo, en [18] se presenta un método para determinar el tamaño de muestra adecuado para una serie de ensayos clínicos para identificar nuevos agentes terapéuticos.

En [19] se muestra un trabajo motivado por un problema específico en experimentos de microarray; el problema de la elección del tamaño de la muestra se plantea como parte de un problema de decisión. En [20] se expone un método que permite el cálculo del número de sujetos requeridos en un estudio de confiabilidad.

El segundo escenario se presenta cuando dado un amplio conjunto de observaciones P acerca de algún fenómeno, es necesario obtener un subconjunto S para inferir su comportamiento. Por razones prácticas (generalmente de tiempo o espacio) P no puede ser usado. Esta situación es común en diferentes enfoques y métodos de MD y Machine Learning (ML) en donde el desempeño es importante. Habitualmente este problema es llamado *instance selection* [21].

Al respecto, varios trabajos han sido publicados. La mayoría de ellos reducen el número de instancias (elementos de P) encontrando un subconjunto de ellas que permita entrenar un clasificador, que determine adecuadamente la etiqueta o clase de las instancias restantes. El clasificador más usado con este propósito es Nearest Neighbor (NN). Dentro de estos trabajos sobresalen: Condensed NN (CNN) [22], Edited NN (ENN) y Repeated Edited NN (ERNN) [23], Variable-kernel Similarity Metric (VSM) [24], Shrink and Growth [25].

Otros enfoques tienen como meta eliminar sistemáticamente elementos de P, dependiendo de la habilidad que tienen los elementos restantes de ser clasificados adecuadamente. Tal es el caso de DROP [26] y el conocido Stratified Ordered Selection (SOS) [27].

Dado que el espacio de solución del problema de *instance selection* puede resultar muy complejo en función de la cardinalidad de P, algunos trabajos basados en heurísticas han surgido. Por ejemplo, en [28] se sugiere un método basado en Random Mutation Hill Climbing (RMHC). En [29] y [30] se aborda el problema usando algoritmos genéticos.

El trabajo aquí presentado propone un método asociado al segundo escenario y va más allá del hecho de encontrar una muestra S^* que optimice el proceso de entrenamiento en tareas de clasificación. El método aquí propuesto puede ser aplicado a problemas de aprendizaje supervisado y no supervisado en los que generalmente se requiere usar conjuntos reducidos de datos.

El problema planteado consiste en encontrar una muestra S de tamaño n, de P, que sea adecuada en función de algún criterio. Se propone como criterio la *cantidad de información* [31]. El objetivo es encontrar un método que más allá de proponer una muestra de tamaño n, nos permita obtener el valor óptimo de n.

La hipótesis es que la optimalidad de n dependerá de la información de S con respecto a P. Esto implica que la cantidad de información de S debe ser

tan similar como sea posible a la de P. Para medir la información, se recurre a la *Teoría de la Información* [32], específicamente al concepto de Entropía.

2.1. Cantidad de información de un conjunto de datos.

En teoría de la información, la entropía es una medida asociada a la *cantidad de información* de una variable aleatoria Y, con posibles valores y_1, y_2, y_r. Desde un punto de vista estadístico, la información del evento $(Y = y_j)$ o simplemente y_j es inversamente proporcional a su probabilidad. Dicha información se denota por $\mathrm{I}(y_j)$ y se puede expresar matemáticamente como:

$$I(y_j) = log\left(\frac{1}{p(y_j)}\right) \tag{2.1}$$

La entropía de Y es el valor esperado de I y está dada por:

$$H(Y) = \sum_{i=1}^{r} p(y_j) log(\frac{1}{p(y_j)}) = -\sum_{i=1}^{r} p(y_j) log(p(y_j)) \tag{2.2}$$

Generalmente la función logaritmo se asume como el logaritmo en base 2, en cuyo caso la entropía queda expresada en bits. También es posible usar In, donde la entropía se expresa en nats. En este trabajo se toma el logaritmo en base 2.

Considerando a P y S como variables aleatorias, se puede calcular el valor de sus entropías a través de (2.2). Se quiere seleccionar una muestra S_i, de tamaño n de P, tal que:

$$\frac{|H(S_i) - H(P)|}{H(P)} \leq \epsilon \tag{2.3}$$

donde ϵ es un parámetro que representa el máximo error permitido entre la información de P y S.

De (2.2) se puede apreciar que la entropía implica determinar las probabilidades $p(y_i)$, por lo tanto se debe conocer la función de densidad de probabilidad (FDP) de Y. Ya que generalmente la FDP es desconocida, se requiere encontrar una aproximación a esta, a través de algún método de inferencia. En este trabajo se usa un enfoque no parámetrico con el fin de evitar asumir una función de densidad en particular. El término no parámetrico no significa ausencia de parámetros sino el poder conservar tan pocos como sea posible. Las aproximaciones no paramétricas pueden incluir funciones de densidad, funciones de densidad condicional, funciones de regresión o funciones basadas en cuantiles para encontrar la distribución más adecuada [33].

Hasta ahora se ha asumido que Y es una variable aleatoria unidimensional. Sin embargo es importante considerar el caso multidimensional, para lo cual se debe medir la entropía de $Y \ \epsilon \ \Re^d$, $d \geq 2$. Esto implica determinar la probabilidad del evento Y= $\overrightarrow{y}$= [y_1, y_2,..., y_d] denotado como $p(y_1, y_2, ..., y_3)$. En la siguiente sección se presenta un método para determinar esta probabilidad.

2.2. Distribución de probabilidad de Y.

Dada una variable aleatoria unidimensional Y, podemos dividir el espacio en un conjunto de intervalos, conocidos como cuantiles [34], los cuales se ilustran en la siguiente figura:

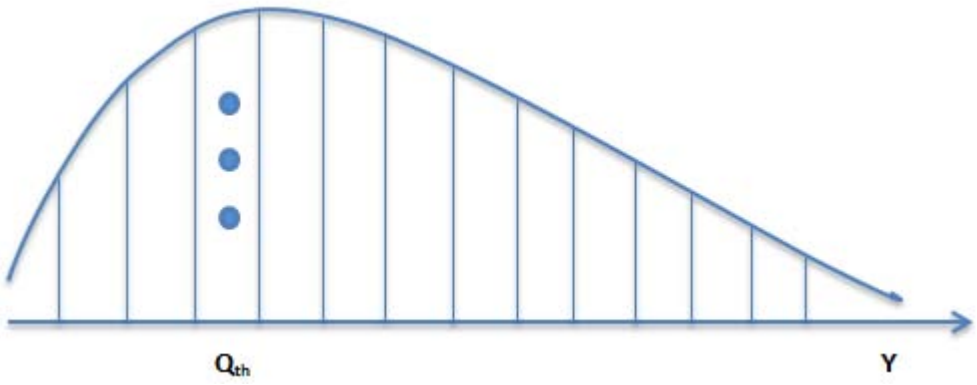

Figura 2.1: División del espacio de Y.

Un cuantil q_i es un intervalo de la forma $q_i = [\ \underline{y}, \overline{y}\ ]$ en donde $\underline{y}$ e $\overline{y}$ son los límites inferior y superior de q_i, respectivamente. La amplitud del cuantil se denota como $\triangle$ y está dada por:

$$\triangle = \frac{|\text{max(Y) - min(Y)}\ |}{m} \tag{2.4}$$

en donde m es el número de cuantiles en que deseamos dividir el espacio. Este valor es definido a priori. El primer cuantil se define como un intervalo semi-cerrado, de la forma:

$$q_1 = [min(Y),\ min(Y) + \triangle) \tag{2.5}$$

Los cuantiles siguientes pueden ser definidos como:

$$q_i = \begin{cases} [\overline{y}_{i-1},\ \overline{y}_{i-1} + \triangle] & si\ i = m \\ [\overline{y}_{i-1},\ \overline{y}_{i-1} + \triangle) & en\ otro\ caso \end{cases} \tag{2.6}$$

donde $\overline{y}_{i-1}$ es el límite superior del cuantil previo. La idea anterior se puede extender a un espacio multidimensional, en el que un cuantil será una partición $d - dimensional$ del espacio de Y. De esta manera, dado un conjunto $Y \epsilon\ \mathbb{R}^d$ con elementos de la forma: $y = [y_1, y_2, ..., y_d]$, podemos dividir su espacio en un conjunto de cuantiles $d - dimensionales$, como se aprecia en la siguiente imagen, para $d = 3$:

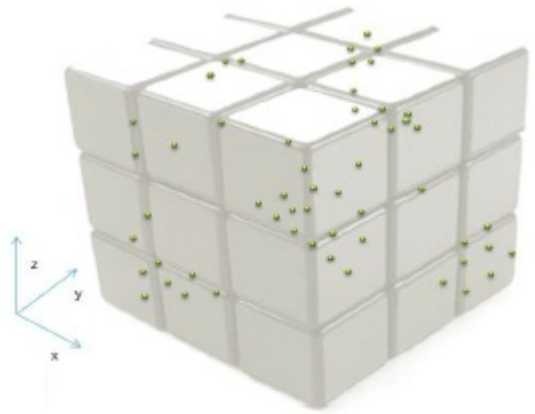

Figura 2.2: División del espacio de Y en 3 dimensiones.

Un cuantil $d-dimensional$ está compuesto por un conjunto de intervalos que determinan los límites superior e inferior para cada dimensión y está definido mediante la siguiente expresión:

$$q_i = [[\underline{y}_{i1},\ \overline{y}_{i1}], [\underline{y}_{i2},\ \overline{y}_{i2}], ..., [\underline{y}_{id},\ \overline{y}_{id}]] \tag{2.7}$$

donde $\underline{y}_{i,k}$e $\overline{y}_{i,k}$ son los límites superior e inferior del cuantil q_i en la $k-ésima$ dimensión. La amplitud de cada intervalo está dada por:

$$\triangle_k = \frac{|max(Y_k) - min(Y_k)|}{m} \tag{2.8}$$

en donde Y_k es el conjunto de datos en la $k-ésima$ dimensión. Con base en lo anterior, podemos generalizar la manera de obtener los límites de un cuantil cuando $Y \epsilon\ \mathbb{R}^d$, como sigue:

$$q_1 = \begin{bmatrix} [min(Y_1),\ min(Y_1) + \triangle_1) \\ [min(Y_2),\ min(Y_2) + \triangle_2) \\ ... \\ [min(Y_d),\ min(Y_d) + \triangle_d) \end{bmatrix}^T \tag{2.9}$$

para el primer cuantil, y para los cuantiles sucesivos mediante la siguiente expresión:

$$q_i = \begin{cases} \begin{bmatrix} [\overline{y}_{(i-1),1},\ \overline{y}_{(i-1),1} + \triangle_1] \\ [\overline{y}_{(i-1),2},\ \overline{y}_{(i-1),2} + \triangle_2] \\ ... \\ [\overline{y}_{(i-1),d},\ \overline{y}_{(i-1),d} + \triangle_d] \end{bmatrix}^T & si\ i = m \\ \\ \begin{bmatrix} [\overline{y}_{(i-1),1},\ \overline{y}_{(i-1),1} + \triangle_1) \\ [\overline{y}_{(i-1),2},\ \overline{y}_{(i-1),2} + \triangle_2) \\ ... \\ [\overline{y}_{(i-1),d},\ \overline{y}_{(i-1),d} + \triangle_d) \end{bmatrix}^T & en\ otro\ caso \end{cases} \tag{2.10}$$

donde $\overline{y}_{(i-1),k}$ es el límite superior del cuantil anterior $(i-1)$ en la $k-ésima$ dimensión. La función de densidad de probabilidad (FDP) de Y está aproximada por la proporción de elementos que pertenecen a cada cuantil. En general, dado una variable aleatoria de la forma $Y = [y_1, y_2, ..., y_d]$ la probabilidad $p(y_1, y_2, ..., y_d)$ es la densidad (en términos de proporción) del cuantil q_i al que el vector $[y_1, y_2, ..., y_d]$ pertenece. Con base en lo anterior, podemos aproximar la FDP de P y S para obtener sus entropías.

2.3. Determinando el número de cuantiles.

Para determinar el valor de m, regularmente se utiliza la fórmula de Sturges [35] [34]. Existen otros métodos que intentan mejorar su desempeño, evitando hacer supuestos de normalidad de los datos como: Doane [36] y Rice Rule [37]. En este trabajo se decidió usar Rice Rule la cual esta dada por:

$$m = \sqrt[3]{2n} \tag{2.11}$$

donde n es el número de observaciones o elementos del conjunto. Para el caso de 1000 elementos, la regla de Sturges sugiere una división del espacio de 11 cuantiles, mientras que Rice Rule recomienda usar 20.

Capítulo 3

Problema de muestreo como problema de optimización.

Como se mencionó, una muestra será mejor que otra en la medida en que su entropía (cantidad de información) sea lo más cercana posible a la entropía de la población. El objetivo es encontrar una muestra S^* de tamaño n (S^*_n), que satisfaga la ecuación 2.3.

En donde ϵ es un umbral de aceptación definido a priori que indica el máximo error permitido entre la entropía de P y la de S^*. El tamaño del espacio de búsqueda de S^* está dado por:

$$\sum_{n=1}^{N} \binom{N}{n} = \sum_{n=1}^{N} \frac{N!}{n!(N-n)!} \tag{3.1}$$

Donde $\binom{N}{n}$ es el número de formas de escoger n elementos de P. Para ilustrar lo anterior, calculamos el coeficiente binomial para $n = 1$ hasta 10, asumiendo una población P cuyo tamaño $N = 1000$.

Los resultados se muestran en la siguiente tabla:

N	n	Número de muestras.
1000	1	1000
	2	499500
	3	166167000
	4	41417124750
	5	8250291250200
	6	1368173298991500
	7	194280608456793000
	8	24115080524699400000
	9	2658017764500200000000
	10	263409560461970000000000
	$\sum_{n=1}^{N} \binom{N}{n}$	266091888964069000000000

Cuadro 3.1: Número de formas de elegir n elementos de P.

Dado que el espacio de búsqueda es vasto y sus características (convexidad, acotamiento, continuidad, etc.) son desconocidas, no es viable el uso de técnicas de optimización tradicionales como programación lineal [38], programación entera [39] o cálculo diferencial [40]. Esto sugiere el uso de alguna técnica meta-heurística [41] que permita explorarlo de manera eficiente.

Entre los métodos existentes se pueden enunciar: búsqueda tabú [42], recocido simulado [43], optimización por colonia de hormigas [44], optimización por enjambre de partículas [45] y cómputo evolutivo [46]. Además, dentro de las variaciones del cómputo evolutivo, encontramos estrategias evolutivas [47], programación evolutiva [48], programación genética [49], y algoritmos genéticos (AGs) [50].

Todos estos métodos se usan para encontrar soluciones aproximadas a problemas de optimización complejos. Se ha demostrado que los algoritmos genéticos con elitismo (aquellos que conservan una proporción de los mejores individuos en cada generación) siempre convergen al valor óptimo global [51]. No obstante, dicha convergencia no está limitada en tiempo, por lo que se necesita elegir un algoritmo genético eficiente que converja a la solución en un tiempo razonable. Por esta razón, se propone el uso de un algoritmo genético (GA) denominado Algoritmo Genético Ecléctico (EGA), el cual demostró ser altamente eficiente para resolver un variado acervo de problemas de optimización, en com-

paración con otros GAs y otras heurísticas.

Los resultados de este estudio se reportan en [52]y[53] . En el Apéndice A: Eclectic Genetic Algorithm y en el Apéndice B: Pseudocódigo del EGA se pueden encontrar más detalles sobre el EGA. Asimismo, en el Apéndice C: EGA Code se puede observar el código que se usó para el desarrollo de experimentos y obtención de resultados de este trabajo.

3.1. Definiendo la función objetivo.

La intención es encontrar el valor mínimo de n que permita obtener una muestra S_i (de tamaño n) de P, cuya entropía sea lo más parecida posible a la de P. Esta función se define en la siguiente fórmula:

$$Minimizar\ f(n) = \frac{n}{N} \tag{3.2}$$

El valor de la función objetivo tiende a 1 cuando el valor de n se aproxima al de N. La función anterior debe satisfacer las siguientes condiciones:

1. El error entre la información de P y S_i debe ser menor o igual a ϵ (ec. 2.3)

2. Como S_i y P son conjuntos de datos tal que $S_i \subseteq P$, el conjunto de cuantiles es único para ambos. Se desea que dados P y S_i, la proporción de elementos de P que caen en el $k-ésimo$ cuantil sea tan similiar como sea posible a la proporción de elementos de S_i que caen en ese mismo cuantil.

Entonces, la función objetivo queda definida como sigue:

$$Minimizar\ f(n) = \frac{n}{N} \tag{3.3}$$

Sujeto a:

$$\frac{|H(S_i) - H(P)|}{H(P)} \leq \varepsilon$$

$$\frac{1}{m} \sum |q_k(S_i) - q_k(P)|^2 \leq \delta$$

$$n \, \epsilon \, [2, N)$$

Donde $q_k(S)$ y $q_k(P)$ son la proporción de elementos de S y P que pertenecen al $k-ésimo$ cuantil, respectivamente. La diferencia entre estas proporciones es una medida de error y se establece que su valor promedio debe ser menor o igual a δ. En este trabajo, se definen $\epsilon = 0.05$ y $\delta = 0.01$, ya que estos valores mostraron una rápida convergencia del proceso de búsqueda a la solución óptima.

Capítulo 4

Metodología.

Como se mencionó, dado el amplio espacio de búsqueda que representa el problema de encontrar la muestra óptima, se utilizará un método heurístico que permita explorar eficientemente dicho espacio. Asimismo, se hará uso de un algoritmo genético denominado EGA, para el cual se ha mostrado estadísticamente que converge eficientemente a la solución de problemas complejos (aquellos con grandes espacios de búsqueda). En las siguiente secciones se mostrará la metodología para solucionar el problema de muestreo planteado como un problema de optimización en el capítulo anterior.

4.1. Codificación del problema.

El EGA propone w soluciones candidatas para obtener las mejores w muestras S_i, de P. Cada candidato se codifica como una cadena binaria de longitud 64. Los 32 bits más significativos codifican un entero sin signo que corresponde al valor de n. Los bits siguientes codifican un entero sin signo que corresponde a la semilla desde la que la muestra S_i fue elegida de P. En la siguiente figura se ilustra esta codificación:

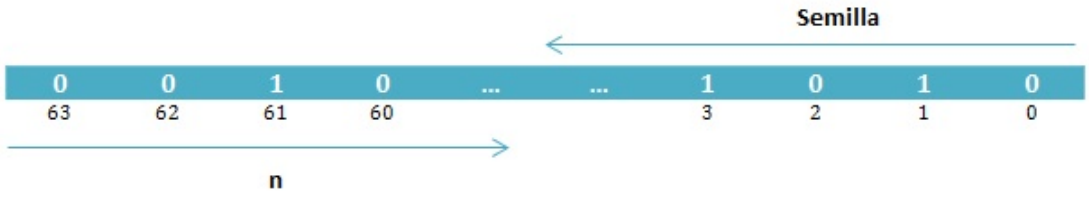

Figura 4.1: Codificación del problema.

Cada solución candidata debe satisfacer las restricciones en (3.3). Cuando una solución candidata no satisface estas condiciones, se castiga mediante una función de penalización (ver 4.1.1).

La evolución se manifiesta después de aplicar varias veces los operadores genéticos del EGA. Cuando se alcanza el máximo número de iteraciones, se tienen w soluciones candidatas que contienen los mejores valores (n y la semilla aleatoria) para obtener S_i. La solución óptima estará en la posición número 1 de la lista de posibles soluciones.

4.1.1. Manejo de restricciones.

La manera más común para resolver problemas de optimización con restricciones es usando una función de penalización [54]. En este caso, la función objetivo puede ser transformada a la siguiente ecuación:

$$f(n) = \begin{cases} f(n) & Si\ n\ es\ una\ solución\ factible. \\ f(n) + penalización(n) & En\ otro\ caso. \end{cases} \tag{4.1}$$

Existen muchas variaciones de la función de penalización. Se decidió usar un método que con base en los resultados de un análisis exhaustivo en [55] mostró tener el mejor rendimiento. Finalmente, la función objetivo queda definida de la siguiente manera:

$$f(n) = \begin{cases} [K - \sum_{i=1}^{s} \frac{K}{p}] & Si\ \ s \neq p \\ f(n) & En\ otro\ caso. \end{cases} \tag{4.2}$$

En donde K es una constante de gran tamaño $[O(10^9)]$, p es el número de restricciones y s representa el número de restricciones que han sido satisfechas.

El valor de K nos permite separar aquellos individuos que satisfacen 1, 2, ..., p restricciones, lo cual favorece a aquellas soluciones que satisfacen el mayor número de restricciones.

4.1.2. Definición de parámetros (Pc, Pm, |C|, G).

Sabemos que el valor de ϵ es definido a priori. Para nuestro proceso evolutivo, se eligió un valor de $\epsilon = 0.01$ y $\delta = 0.01$. Además se definieron los siguientes parámetros asociados al EGA:

Parámetro	Descripción	Valor
P_c	Probabilidad de cruzamiento	0.90
P_m	Probabilidad de mutación	0.01
$\|C\|$	Número de candidatos	80
G	Número de iteraciones	100

Cuadro 4.1: Parámetros del proceso evolutivo.

Estos valores se basan en un estudio mencionado en [56], que muestra desde el punto de vista estadístico, que el EGA converge a la solución óptima alrededor de ellos, cuando se trabaja con problemas de alta exigencia (con un amplio y no convexo espacio de búsqueda factible).

4.2. Obtención de la muestra óptima.

Una vez que el proceso evolutivo del EGA finaliza, la población contiene al mejor candidato, cuyo genoma incluye el valor óptimo de la muestra y una semilla aleatoria, denotados como n^* y r, respectivamente. Esto nos permite conocer la muestra óptima S^* de tamaño n^* extraída de P.

El proceso comprende los siguientes pasos:

1. Generar un conjunto aleatorio C de soluciones candidatas de acuerdo a la codificación del problema.

2. Para cada candidato $c_i \epsilon C$, decodificar su genoma a fin de obtener un valor de n y r.

3. Obtener una muestra aleatoria S_i de tamaño n de P, usando r.

4. Determinar el valor de rendimiento de c_i con base en la función objetivo (ec. 3.3)

5. Determinar la factibilidad de c_i y aplicar la función de penalización (ec. 4.2), en caso de ser necesario.

6. Ordernar C de manera ascendente, basado en los valores de rendimiento.

7. Aplicar los operaores genéticos del EGA (ver Apéndice A: Eclectic Genetic Algorithm).

8. Repetir los pasos 2-7 hasta que se cumpla el criterio de convergencia (usualmente un número de generaciones asignado).

9. Seleccionar al elemento que se encuentra en la posición número uno en la lista de candidados de C, decodificarlo para de obtener un valor de n y r.

10. Obtener una muestra de tamaño $n*$ usando r. Esta muestra será la muestra óptima denotada como S_i^*.

Capítulo 5

Resultados obtenidos y aplicaciones.

En este capítulo se muestran los resultados obtenidos con el método propuesto. En primer lugar se exponen una serie de resultados preliminares que permitieron dar evidencia de su efectividad. Para dar certeza estadística a dicha efectividad se proponen subsecuentemente dos métricas de evaluación:

- a) Porcentaje de reducción de datos (reducción del tamaño de S^* con respecto a P).

- b) El error de muestreo (diferencia de las propiedades estadísticas de S^* con respecto a P) .

El promedio de estas métricas fue calculado a través de conjuntos de datos sintéticos sobre los cuales se aplicó el método planteado. Generalmente los conjuntos de datos poseen información "oculta" que no es posible caracterizar a través de medidas estadísticas. Se busca que este método también garantice que el subconjunto S^*conserve esta información. Para ello, se recurre a la capacidad de clasificación de S^* respecto a P. En este sentido, se usó una serie de conjuntos de datos que representan problemas de clasificación en donde la marca de clase es conocida (*labeled dataset*). Se espera que dado un gran conjunto de datos etiquetado P_l, se pueda encontrar un subconjunto S_l^* que sea capaz de entrenar un clasificador, en lugar de usar P_l. La efectividad está dada por el porcentaje de coincidencia entre las etiquetas encontradas por el clasificador entrenado con S_l^* y las obtenidas por el mismo clasificador pero entrenado con P_l. (Subsección 5.3).

Por último, se expone la efectividad del método con el uso de conjuntos de datos en un contexto de aplicación.

5.1. Resultados preliminares.

Los siguientes experimentos mostraron resultados que revelan que el método propuesto es prometedor.

5.1.1. Conjuntos de datos unidimensionales.

- Conjunto de datos C_1 de 10000 elementos extraído de una distribución gaussiana con parámetros $\mu = 10$ y $\sigma = 2$.

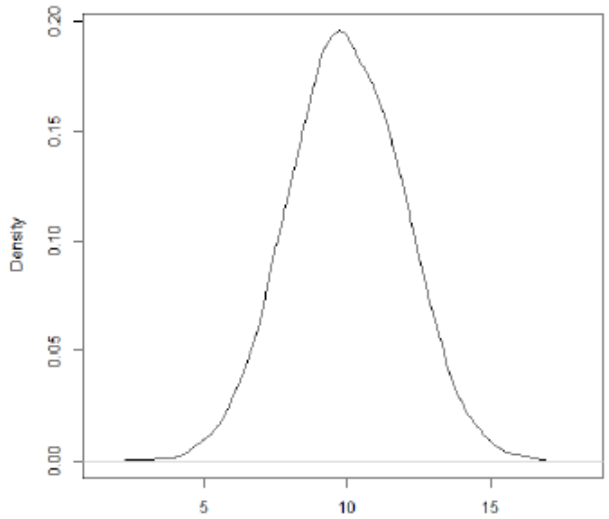

Figura 5.1: Densidad de C_1.

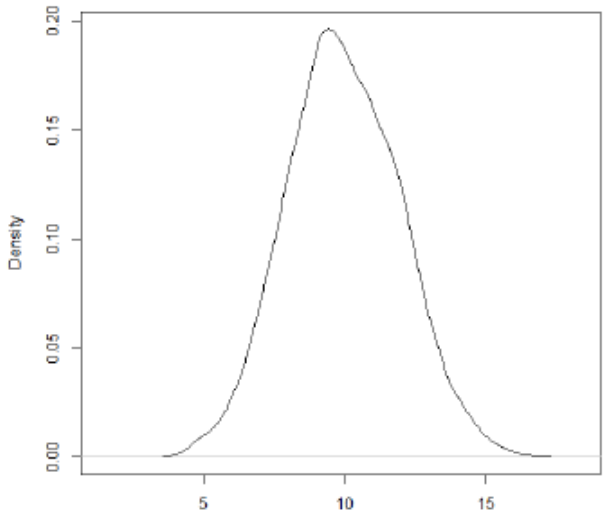

Figura 5.2: Densidad de S_1^* obtenida a partir de C_1 ($n* = 144$).

- Conjunto de datos C_2 de 10000 elementos extraído de una distribución de Poisson con parámetro $\lambda = 3$.

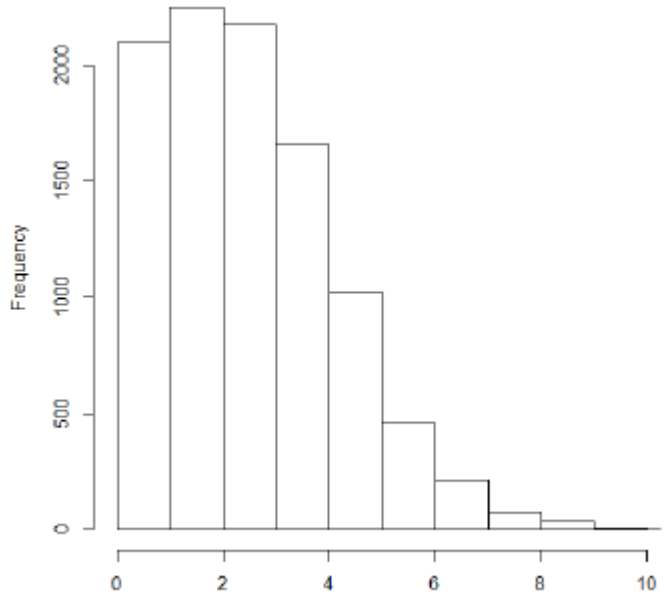

Figura 5.3: Histograma de C_2.

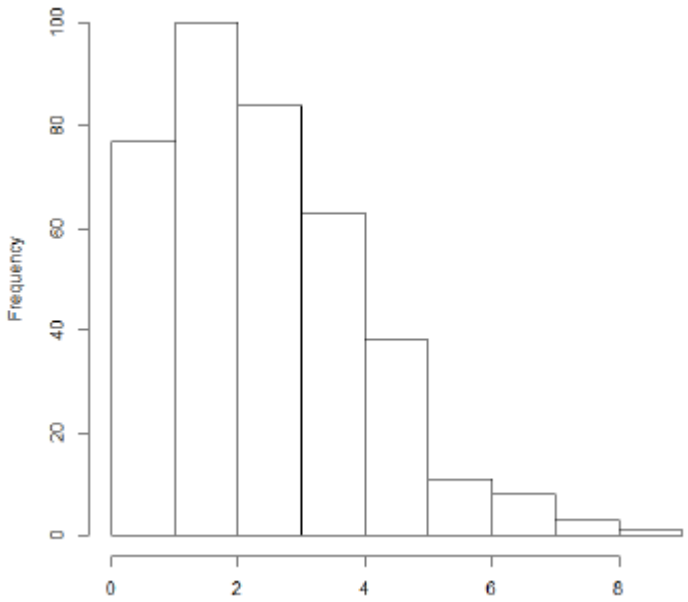

Figura 5.4: Histograma de S_2^* obtenida a partir de C_2 ($n* = 385$).

- Conjunto de datos C_3 de 10000 elementos extraído de una distribución de Weibull con parámetro $\lambda = 1$ (escala) y $k = 1.5$ (forma) .

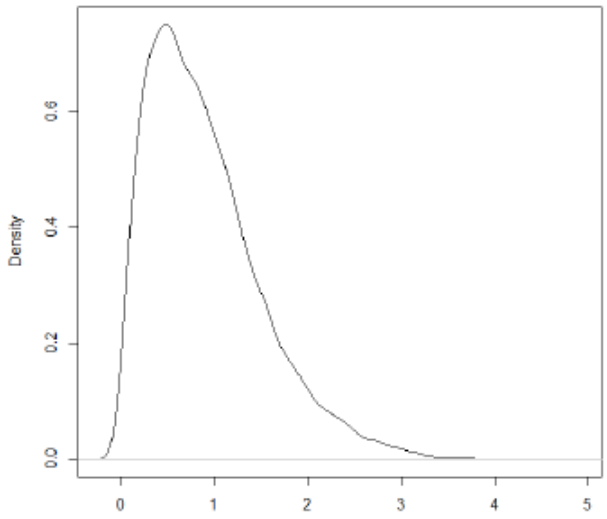

Figura 5.5: Densidad de C_3.

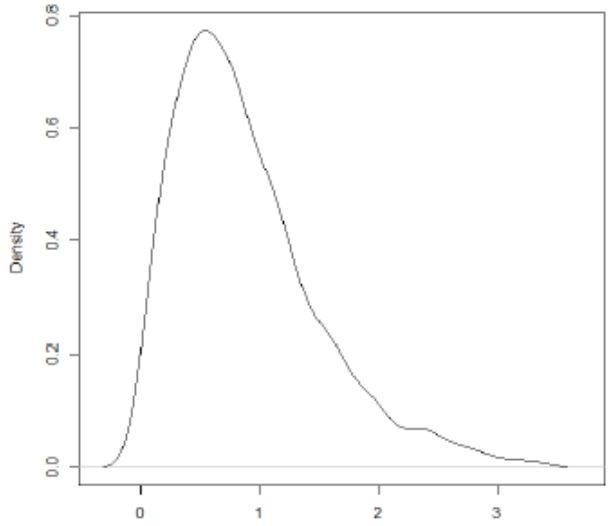

Figura 5.6: Densidad de S_3^* obtenida a partir de C_3 ($n* = 1120$).

Los experimentos muestran algunas evidencias respecto a la hipótesis de que el método propuesto obtiene una muestra óptima S^* que mantiene las propiedades estadísticas dadas por la FDP, debido a que dichas muestras conservaron propiedades como unimodalidad, sesgo y curtosis, como se puede apreciar en las gráficas.

5.1.2. Conjuntos de datos bidimensionales.

- Conjunto de datos C_4 de 10000 elementos extraído de una distribución gaussiana con parámetros $\overrightarrow{\mu} = [0.5, 0.5]$ y $\overrightarrow{\sigma} = [1.0, 1.0]$.

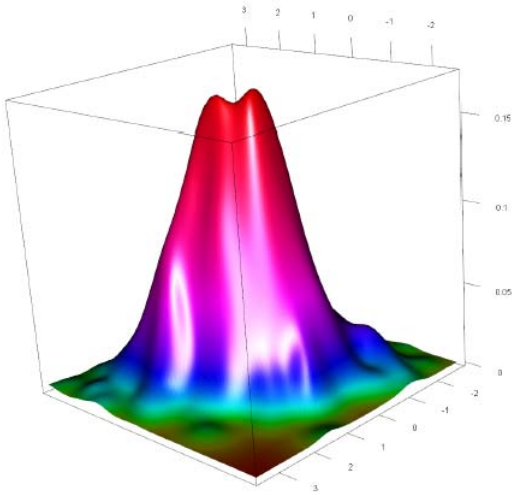

Figura 5.7: Densidad de C_4.

En este caso se encontró una muestra S_4^* con los siguientes parámetros: $\overrightarrow{\mu} = [0.51, 0.48]$ y $\overrightarrow{\sigma} = [0.98, 0.96]$. Con lo cual podemos apreciar que estos valores son muy cercanos a los originales del conjunto C_4.

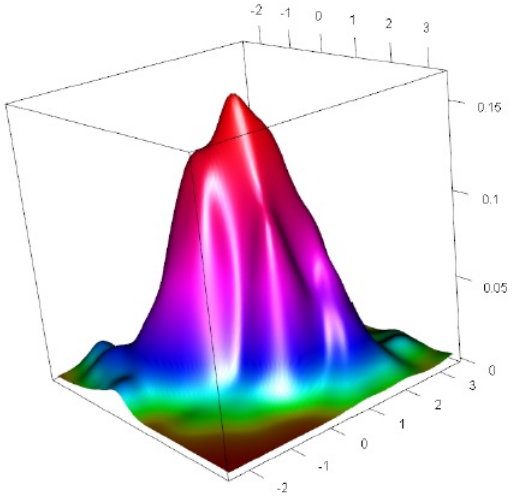

Figura 5.8: Densidad de S_4^* obtenida a partir de C_4 ($n* = 642$).

- Conjunto de datos C_5 de 10000 elementos que representan una función sinusoidal dentro del intervalo $[-2\pi, 2\pi]$.

Como la distribución del conjunto de datos sinusoidal no está caracterizada por una FDP, se estiman las distribuciones marginales y se comparan contra las distribuciones marginales de S_5^*. En las siguientes gráficas se puede apreciar las distribuciones marginales de C_5 y S_5^* en la abscisa (x) y en la ordenada (y).

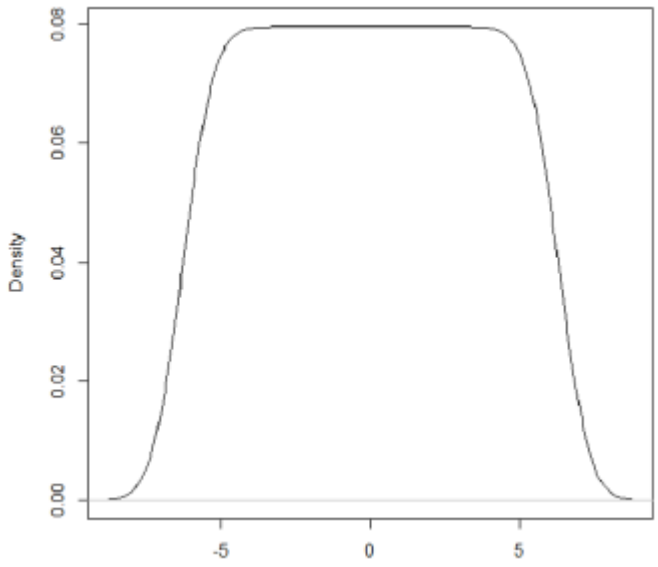

Figura 5.9: Distribución marginal de C_5 en x.

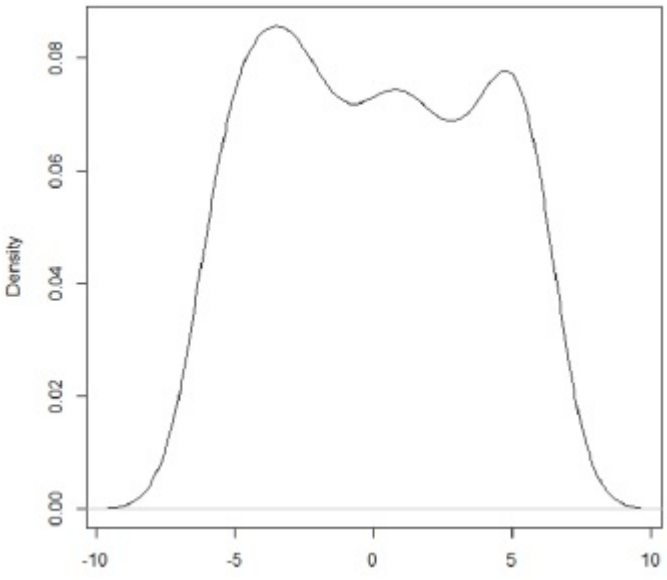

Figura 5.10: Distribución marginal de S_5^* en x ($n* = 642$).

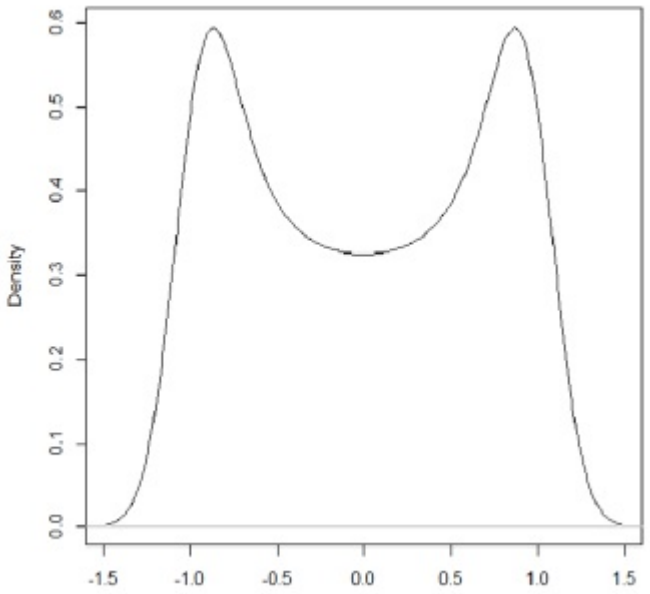

Figura 5.11: Distribución marginal de C_5 en y.

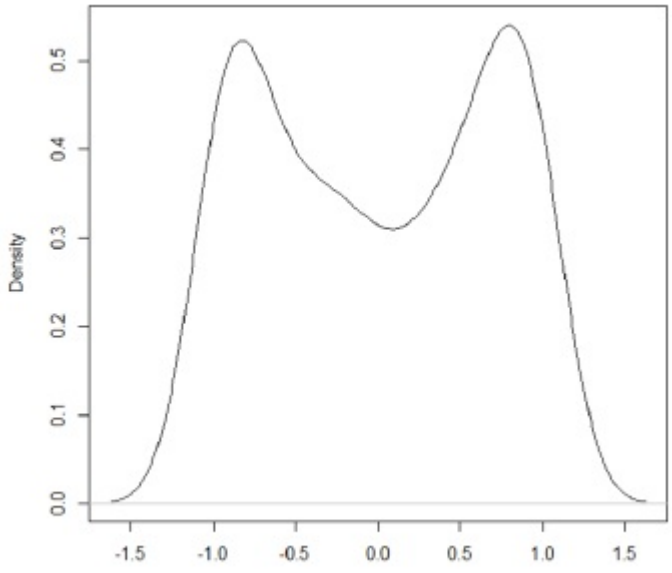

Figura 5.12: Distribución marginal de S_5^* en y ($n* = 642$).

En el caso de y la similitud entre las distribuciones es evidente. No obstante, las distribuciones en x muestran una ligera diferencia. Debido a esto, se analizaron los datos calculando su distribución conjunta, como se muestra a continuación:

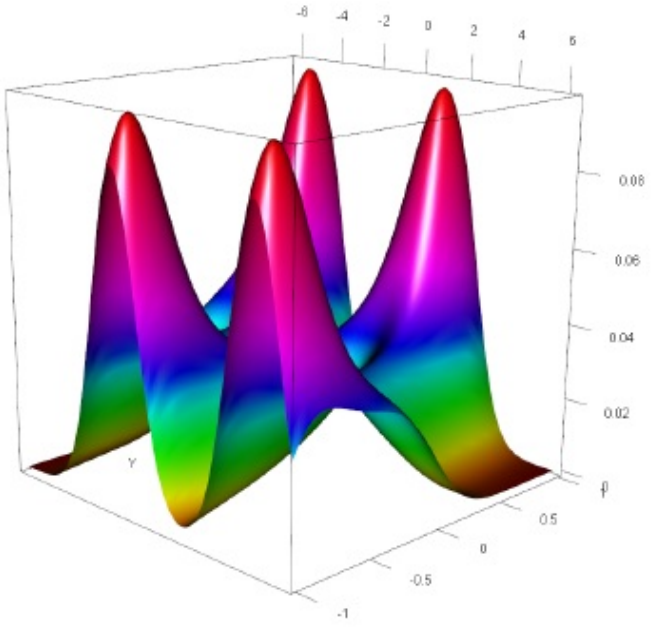

Figura 5.13: Distribución conjunta de C_5.

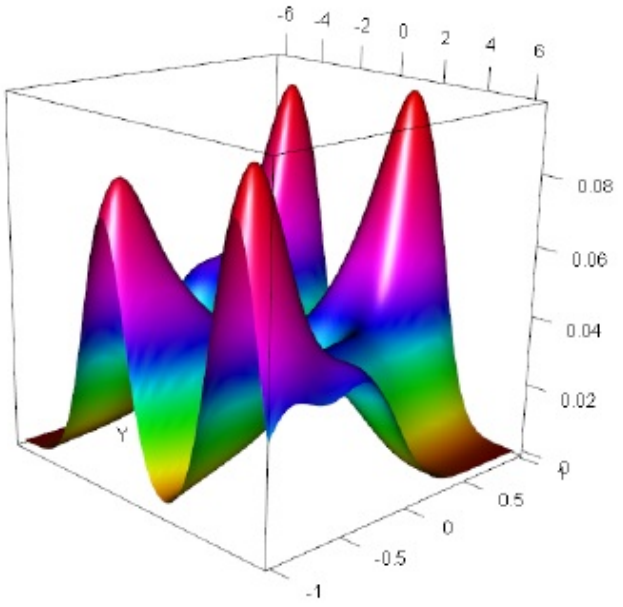

Figura 5.14: Distribución conjunta de S_5^* ($n* = 642$).

Los resultados anteriores muestran que las muestras logran conservar la información emitida por el conjunto de datos original y reducir de manera significativa su tamaño.

Sin embargo, estos resultados no son suficientes para generalizar esta

observación. Por lo tanto, en las siguientes secciones se presenta una metodología experimental que permite generalizar la eficiencia del método propuesto.

5.2. Evaluación del desempeño.

Para medir el desempeño del método, se utilizó una métrica llamada space saving (SS) que sirve para establecer el nivel de compresión de datos. Dicha métrica se encuentra definida por la siguiente ecuación:

$$SS = 1 - \frac{N}{|P|} \tag{5.1}$$

Un valor grande de SS; próximo a 1, implica un mejor rendimiento. Se calculó esta métrica con un amplio conjunto de experimentos, alrededor de 5000, los cuales incluyeron conjuntos de datos de cardinalidad 1000, 5000, 10000 y 100000. Se determinó el intervalo de confianza de los resultados, con un valor de 0.05. Tales resultados se presentan en la siguiente imagen.

SS	0.7895
Desviación estándar	0.3181
Límite inferior	0.7806
Límite superior	0.7983
Nivel de confianza	95 %

Cuadro 5.1: Efectividad de la reducción de datos.

Los resultados muestran que en promedio el proceso de muestreo permite reducir el tamaño del conjunto de datos hasta en un 70 %. Generalmente las propiedades estadísticas que se deben preservan en el conjunto de datos reducido pueden ser caracterizadas por los siguiente estadísticos: μ y σ. Por lo tanto, se hace uso de ellos para definir las siguientes medidas de desempeño:

$$\begin{aligned} error_{\mu} &= \| \vec{\mu}_{S^*} - \vec{\mu_P} \| \\ error_{\sigma} &= \| \vec{\sigma}_{S^*} - \vec{\sigma}_P \| \end{aligned} \tag{5.2}$$

donde $\vec{\mu}_{S^*}$, $\vec{\sigma_{S^*}}$, y $\vec{\mu}_P$, $\vec{\sigma_P}$ son el vector de medias y el vector de desviaciones estándar de S^*y P, respectivamente.

Los términos $error_\mu$ y $error_\sigma$ representan una norma euclidiana. Dado que puede haber conjuntos de datos con diferentes escalas, las normas obtenidas con ellos podrían resultar no comparables. Para evitar este problema, se escalaron los datos en P y S^* entre 0 y 1. En consecuencia, un valor pequeño de $error_\mu$ y $error_\sigma$ (cercano a 0) implica un mejor rendimiento.

Durante la ejecución de los experimentos para obtener el valor promedio de SS, los valores de $error_\mu$ y $error_\sigma$ también fueron calculados. Dichos valores se exponen enseguida:

	$error_\mu$	$error_\sigma$
Promedio	0.016	0.021
Desviación estándar	0.00	0.031
Límite inferior	0.00	0.031
Límite superior	0.00	0.031
Nivel de confianza	95 %	95 %

Cuadro 5.2: Error de muestreo caracterizado.

Estos resultados revelan que en general S^*mantiene alrededor del 98 % de las propiedades estadísticas de P.

Hasta ahora se ha mostrado que el método puede reducir el espacio del conjunto de datos en más de un 70 % y preservar las propiedades estadísticas. Sin embargo, regularmente los conjuntos de datos contienen información escondida cuyo reconocimiento es difícil a traves de métodos estadísticos.

Se quiere garantizar que el subconjunto S^* también conserve dicha información. Se discute al respecto en las siguientes secciones.

5.3. Conservando información oculta.

A fin de ilustrar la efectividad del método, de mantener información oculta cuando se muestrea un conjunto de datos, se seleccionaron 5 cojuntos de datos; Abalone [57] , Cars [58], Census Income [59], Hepatitis [60] y Yeast [61], del repositorio UCI Machine Learning, cuyas propiedades se muestran a continuación:

Nombre del conjunto de datos	Variables	Clases	Tamaño	Valores faltantes
Abalone	8	29	4177	No
Cars	22	4	1728	No
Census Income	14	2	32561	Sí
Hepatitis	20	2	155	Sí
Yeast	10	8	8	No

Cuadro 5.3: Propiedades de los conjuntos de datos.

Se eligieron conjuntos de datos etiquetados que representan problemas de clasificación. El criterio de selección de estos conjuntos de datos, estuvo basado en las siguientes características:

- Multidimensionalidad.
- Cardinalidad.
- Complejidad.
- Datos categóricos.
- Datos con valores faltantes.

Algunas de estas características incluyen tareas de preprocesamiento que garantizan la calidad del conjunto de datos. Se aplicaron las siguientes técnicas de preprocesamiento:

- Las variables categóricas fueron codificadas usando variables dinarias artificiales [62].
- Los conjuntos de datos fueron escalados dentro de $[0, 1)$.

- Para completar información faltante, se interpolaron los valores desconocidos con splines naturales [63].

Con estos conjuntos y con base en un clasificador bayesiano (CB), se calculó la efectividad de la siguiente manera:

1. Dado un conjunto de datos etiquetado P_l, obtener un conjunto de prueba no etiquetado, denotado como P_{test}.

2. Obtener la muestra óptima S_l^* de P_l.

3. Ejecutar el proceso de entrenamiento del CB con S_l^*.

4. Ejecutar el proceso predictivo del CB usando P_{test} con el objetivo de obtener un vector etiquetado $\vec{C}$.

5. Ejecutar el proceso de entrenamiento del CB con P_l.

6. Ejecutar el proceso predictivo del CB con P_{test} a fin de obtener un vector etiquetado $\vec{C}'$.

7. Calcular la razon de coincidencia entre $\vec{C}'$ y $\vec{C}$.

Con base en lo anterior, se calculó la efectividad de cada conjunto de datos. Los resultados, así como el índice de reducción de datos (SS) se muestran en la siguiente tabla:

Nombre del conjunto de datos	% de coincidencia	SS
Abalone	0.88	0.63
Cars	0.89	0.71
Census Income	0.94	0.76
Hepatitis	0.87	0.61
Yeast	0.92	0.71
Promedio	0.92	0.68

Cuadro 5.4: Resultados.

La tabla anterior revela que además de reducir el tamaño de los datos, el método propuesto también preserva información oculta (que no puede ser caracterizada estadísticamente) que es necesaria para el proceso de clasificación.

Se puede observar que la razón de coincidencia es de más del 90 %, esto significa que la muestra obtenida permite entrenar un CB tan bien como lo haría el conjunto de datos original. Además, fue posible reducir el tamaño de los conjuntos de datos en un 68 % en promedio.

5.4. Conjuntos de datos con aplicación en un problema real.

5.4.1. Problema de clasificación de caracteres.

Este problema consiste en clasificar de manera adecuada los dígitos 4 y 9. El conjunto de datos se obtuvo del repositorio UCI [64] y contiene 6000 elementos y 5000 atributos. El objetivo de aplicar el método es reducir el número de atributos, por lo tanto, se ejecutaron 6000 mil experimentos, ya que para cada elemento se requería saber cuáles eran los atributos que aportaban mayor información.

Después de llevar a cabo los experimentos, se encontró la frecuencia relativa de cada atributo y se eligió a aquellos que tuvieran una frecuencia relativa mayor a 0.15, con esto se pudo reducir el número de atributos a 890. Para verificar que la reducción mantenía las propiedades del conjuto de datos original (total de atributos), se aplicó una red perceptrón multi-capa (MLPN) con el conjunto de datos reducido. Finalmente el conjunto de etiquetas obtenidas se comparó con el obtenido por el conjunto de datos original y se descubrió una proporción de elementos idénticos (etiquetas) del 97.68 % entre ambos.

Conclusión.

Se ha definido un nuevo método de muestreo, basado en el concepto de Entropía de Shannon, a fin de encontrar una muestra mínima que conserve la información de un conjunto de datos. Encontrar la muestra óptima involucra un problema de optimización que requiere de un procedimiento eficiente para explorar un amplio espacio factible. Se usó el EGA como la mejor alternativa. Una primera aproximación permite verificar que el método es capaz de extraer una muestra que mantiene las propiedades originales relacionadas a la distribución de probabilidad o disposición espacial. Con base en dichos resultados, se ejecutaron varios experimentos sobre conjuntos de datos sintéticos. Se encontró que, en general, la muestra conserva alrededor de un 98 % de las propiedades estadísticas de P y que su tamaño es aproximadamente del 30 % del tamaño del conjunto de datos original. Debido a que frecuentemente los conjuntos de datos poseen información oculta que no es posible caracterizar a través de métodos estadísticos, se quiso asegurar que el conjunto de datos reducido incluye dicha información, para ello, se recurrió a la capacidad de clasificación de la muestra respecto a la de la población. Se utilizó una serie de conjuntos de datos que representan problemas de clasificación, a fin de mostrar que el método obtiene la muestra óptima capaz de entrenar un clasificador bayesiano tan bien como lo harían los datos originales.

En promedio, se alcanzó un valor de SS de 0.73, lo cual implica que se logra reducir la cardinalidad de la población en más de un 70 %, lo que se traduce en un beneficio tanto en tiempo de cómputo necesario para su procesamiento como en el ámbito monetario. Este hecho, deja inferir muchas aplicaciones que requieren compresión de datos sin pérdida, en áreas como la estadística, Machine Learning y Data Mining. Finalmente, se mostró que el método puede ser aplicado a aquellos problemas que exigen reducción de dimensionalidad o selección de características y que podría evitar inducir erroes en el análisis e inferencia de los datos. Como trabajo futuro, se espera mostrar que el método puede ser aplicado a aquellos problemas relacionados con selección de atributos en donde es obligatoria la eliminación de datos redundantes.

Glosario.

Datos: Representación simbólica de atributos o variables que describen entidades, sucesos o fenómenos del "mundo real".

Información: Serie de símbolos que conforman un mensaje, el cual es originado por una *fuente* y recibido por un *receptor* a través de un *canal*.

Cleaning: Técnica de pre-procesamiento de datos que se aplica para remover ruido y corregir inconsistencias en ellos.

Integration: Técnica de pre-procesamiento de datos que fusiona aquellos que provienen de diferentes fuentes, regularmente en data-warehouses.

Data-warehouses: Repositorios de información recopilada de varias fuentes, almacenadas en un esquema unificado, que habitualmente residen en un solo sitio.

Complejidad: Estudia el costo de la solución de problemas en función de la cantidad de recursos de tiempo y espacio necesarios.

Población: Conjunto finito o infinito de personas u objetos que presentan características comunes.

Muestra: Subconjunto representativo de la población a partir del cual se pretende realizar inferencias respecto a la población de donde procede.

Teoría de la información: Rama de las matemáticas aplicadas que cuantifica y analiza la información (desde el punto de vista estadístico), que involucra procesos de almacenamiento, transmisión y recuperación de la misma.

Algoritmo Genético: Heurística de búsqueda global que se utiliza para encontrar soluciones aproximadas a problemas de búsqueda y optimización.

Apéndice A: Eclectic Genetic Algorithm.

El algoritmo genético tradicional [65] usado frecuentemente en la literatura, deja a decisión del investigador los valores de los siguientes parámetros:

1. Probabilidad de cruzamiento (P_c).

2. Probabilidad de mutación (P_m).

3. Cardinalidad de la población.

Además, es de vital importancia la convergencia lenta y/o prematura. Para ello, el EGA incluye las siguientes características:

- Se considera a los mejores n individuos de todo el conjunto de posibles soluciones. El mejor y el peor individuo $(1 - n)$ son seleccionados, después el segundo mejor y segundo peor individuo $(2 - [n - 1])$ son seleccionados, etc.

- Se ejecuta el cruzamiento con una probabilidad P_c. El cruzamiento anular vuelve independiente la posición de esta operación. En él, se eligen dos individuos al azar, estos cromosomas no se consideran como cadenas sino como anillos en el que el último bit del genoma está conectado al primero. Se eligen semi-anillos del mismo tamaño y se intercambia la información genética de ambos semi-anillos, formando nuevos descendientes.

  El cruzamiento anular se ilustra en la siguiente figura:

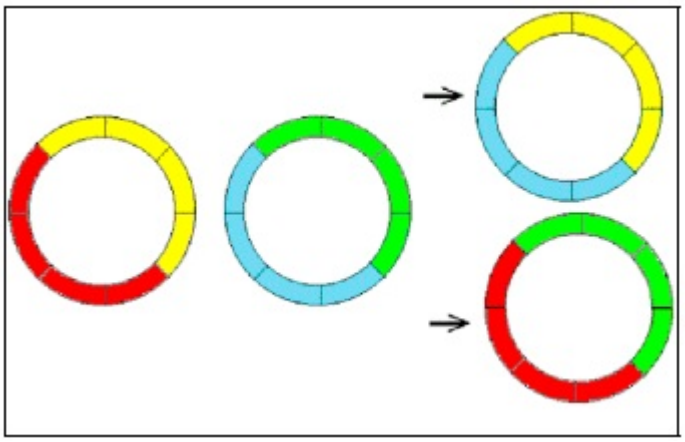

Figura 5.15: Cruzamiento anular.

- A diferencia del operación de mutación tradicional, en el EGA no se consideran probabilidades de mutación para cada bit, en lugar de eso, se toma el valor esperado de las mutaciones, lo cual es equivalente a calcular la probabilidad de mutación para cada bit.

  El valor esperado de las mutaciones se calcula a través de la siguiente expresión: $l * n * p_m$. Donde l es la longitud del individuo expresada en bits, n representa el número de individuos en la población y p_m la probabilidad de mutación.

Apéndice B: Pseudocódigo del EGA.

Algorithm 5.1 EGA.

n= Número de individuos.
p_c= Probabilidad de cruzamiento.
p_m= Probabilidad de mutación.
l = Longitud del individuo.
$b2m = l * m * p_m$ Número de bits a mutar.

Resultado: Los mejores n individuos.
Generar una población P de tamaño n cuyos nl bits son establecidos al azar.
$inicializar(P)$;
$evaluar(P)$;
Ordenar a los individuos del mejor al peor con base a su rendimiento en la función objetivo.
$ordenar(P)$;

while criterio de convergencia no se cumpla **do**
  $duplicar(P)$;
  for $i = 1$ to n **do**
    Generar un número aleatorio R;
    if $R > p_c$ then
      Generar un entero aleatorio $(locus)\ \epsilon\ [1, l]$;
      Intercambiar el semi-anillo comenzando en *locus* para los individuos i e $2n - i + 1$;
      $cruzamiento(P(i),\ P(2n - i + 1))$;
    end
  end
  Mutar $b2m$ bits de la población, seleccionados de manera aleatoria.
  $mutar(P)$;
  $ordenar(P)$;
  Eliminar los peores n individuos de P.
  Regresar P.
end

Apéndice C: EGA Code.

```
################################################################################
#Package Name: Eclectic Genetic Algorithm.
#Package Description: Implements the Eclectic Genetic Algorithm.
#Version 1.0:
#    1. Includes the functions to solve the problem of finding
#    the optimal value of a dataset.
#    2. Includes some sentences to determine the values of
#    the sample based on normality assumption, in order to
#    compare the results of EGA against such traditional
#    method.
################################################################################

################################################################################
# 1. Preliminary Elements.
################################################################################

rm(list=ls(all=TRUE))
library("stringr")
source("Parameters.r") #Parameters File

################################################################################
# 2. Functions of the Eclectic Genetic Algorithm.
################################################################################

################################################################################
#Description: Allows to create the initial population.
################################################################################

initialize<-function()
{
  P<<-matrix(sample(0:1,n*length,TRUE,c(0.5,0.5)),nrow=n)
  F<<-matrix(0.0, nrow=n, ncol=1)
}

################################################################################
#Description: Allows to decode the genome of  an individual.
################################################################################

decode<-function(genome)
{    fenotype=vector(mode="numeric", length=numberOfVariables)
     I=toString(genome)             #String representation of
     ↪  the genes of i-th individual
     I=str_replace_all(I,",","")    #Delete the character ","
     I=str_replace_all(I," ","")    #Delete the blank spaces " "
     strI=paste(I,collapse="")
     k=ifelse(unsigned==0, 0, 1)
```

```
      r=1
      while(r<=numberOfVariables)
      {
         strSignBit=ifelse(unsigned==0,"",substr(strI, k, k))
         strInteger=substr(strI, k+1, integerPart+k)
             if(integerPart<variableLength)
             {
                 strDecimal=ifelse(unsigned==0,substr(strI,
                  ↪  k+(integerPart+1),(variableLength+k)),
                  ↪  substr(strI,
                  ↪  k+(integerPart+1),(variableLength+k)-1))
             }
             else
             {
                 strDecimal=""
             }

         tmpVector=unlist(strsplit(strDecimal, split=""))
         decimalValue=0
         j=1
             while(j<=length(tmpVector))
             {
                 decimalValue=decimalValue+
                  ↪  strtoi(tmpVector[j])*2^-j
                 j=j+1
             }
             signValue=ifelse(strSignBit=="0"|strSignBit=="",1,-1)
             fenotype[r]=(strtoi(strInteger,base=2) +
              ↪  decimalValue)*signValue
             k=k+(variableLength)
             r=r+1
      }
      return(fenotype)
}

#############################################################################
#Description: Allows to decode the population.
#############################################################################

decodePopulation<-function()
{
       i=1
       sizes=vector(mode="numeric", length=n)
       while(i<n)
       {
           sizes[i]=decode(P[i,])
```

```
87            i=i+1
88        }
89        return(sizes)
90
91  }
92
93  ##############################################################################
94  #Description: Allows to calculate the fitness value for each
    ↪  individual.
95  ##############################################################################
96
97  evaluate<-function(from,to)
98  {
99    source("FunctionsSet.R")
100   I             # Vector individual genome
101   strI=""       # String representation of the individual genome
102   strInteger=""
103   strDecimal=""
104   strSignBit=""
105   fenotype=vector(mode="numeric", length=numberOfVariables)
106   i=from
107     while(i<=to)
108       {
109          fenotype=decode(P[i,]) #String representation of the
              ↪  genes of i-th individual
110          F[i]<<-c(evaluateFunction(functionNumber,fenotype))
111          i=i+1
112       }
113 }
114
115 ##############################################################################
116 #Description: Allows to create the initial population.
117 ##############################################################################
118
119 sortPopulation<-function()
120 {
121   indices=order(F)
122   P<<-P[indices,]
123   F<<-matrix(F[indices]) #This changes the dimension of the
      ↪  vector
124 }
125
126 ##############################################################################
127 #Description: Allows to duplicate the population.
128 ##############################################################################
129
```

```
duplicatePopulation<-function()
{
  P<<-rbind(P,P)
  F<<-rbind(F,F)  #It is compulsory also duplicate the vector
   ↪  fitness
}

################################################################################
#Description: Allows to crossover the population.
################################################################################

crossoverPopulation<-function()
{
#Annular crossover
L=length
L_2=L/2
i=n+1
    while(i<=2*n)
    {
      a<-runif(1, 0, 1)
       if (a<pc)
       {
         r<-round(runif(1, L_2+1, L))
         rightIndA<-P[i,r:L]
         leftIndA<-P[i,1:(L_2-length(rightIndA))]
         mediumIndA<-P[i,(length(leftIndA)+1):(r-1)]
         bottom_i=((2*n)-i)+1
         rightIndB<-P[bottom_i,r:L]
         leftIndB<-P[bottom_i,1:(L_2-length(rightIndB))]
         mediumIndB<-P[bottom_i,(length(leftIndB)+1):(r-1)]
         P[i,]<<-c(leftIndB,mediumIndA,rightIndB) #Interchange
          ↪  the left
         P[bottom_i,]<<-c(leftIndA,mediumIndB,rightIndA)


       }
        i=i+1
    }
}

################################################################################
#Description: Allows to mutate the population.
################################################################################

mutatePopulation<-function()
{
```

```
 i=1
 while(i<=b2m)
 {
      a<-round(runif(1, n+1, 2*n))
     b<-round(runif(1, 1, length))

     P[a, b]<<-abs(P[a,b]-1)
     i=i+1
 }
}

#############################################################################
#Description: Allows to remove the worst n indivuals from P.
#############################################################################


removeWorst<-function()
{
 P<<-P[1:n, ]
 F<<-matrix(F[1:n])

}

#############################################################################
#Description: Allows to execute the evolutionary process.
#############################################################################

run<-function()
{
 i=1
 while(i<=g)
 {
    duplicatePopulation()
    crossoverPopulation()
    mutatePopulation()
    evaluate(n+1,2*n)
    sortPopulation()
    removeWorst()

    graph=paste("graph_",i,".png",sep = "")
    png(file.path(getwd(),graph))
    sizes=decodePopulation()
    plot(F,sizes,xlim=c(0,1),ylim=c(0,1000),type="p",
     ↪  col="2",xlab="fitness", ylab="size")
    title(main = "Sample Size vs. Fitness", sub =
     ↪  paste("Generation ",i))
```

```
    grid(20, 20 , col = "lightgray", lty = "dotted", lwd =
    ↪  par("lwd"), equilogs = TRUE)
    dev.off()
    print(paste("The best, iteration",i,":",F[1]))
    fitnessResults[i]=F[1]
    i=i+1
 }
}

########## Execute the evolutionary process ####################

executeGA<-function()
{
    set.seed(SEED_EVOLUTIONARY_PROCESS)    #seed for the
    ↪  evolutionary process
    initialize()
    evaluate(1,n)
    sortPopulation()
    run()
    return(decode(P[1,]))
}

############### End of evolutionary process ####################
```

Apéndice D: Paper.

Metadata of the chapter that will be visualized in SpringerLink

Book Title	Advances in Artificial Intelligence and Soft Computing	
Series Title		
Chapter Title	Finding the Optimal Sample Based on Shannon's Entropy and Genetic Algorithms	
Copyright Year	2015	
Copyright HolderName	Springer International Publishing Switzerland	
Corresponding Author	Family Name	**Aldana-Bobadilla**
	Particle	
	Given Name	**Edwin**
	Prefix	
	Suffix	
	Division	Facultad de Ingeniería UNAM
	Organization	Ciudad Universitaria
	Address	México D.F, Mexico
	Division	
	Organization	Facultad de Estudios Superiores-UNAM
	Address	Naucalpan, Estado de México, Mexico
	Email	edwynjavier@yahoo.es
Author	Family Name	**Alfaro-Pérez**
	Particle	
	Given Name	**Carlos**
	Prefix	
	Suffix	
	Division	Facultad de Ingeniería UNAM
	Organization	Ciudad Universitaria
	Address	México D.F, Mexico
	Division	
	Organization	Facultad de Estudios Superiores-UNAM
	Address	Naucalpan, Estado de México, Mexico
	Email	carlos.alfaro26@gmail.com

Abstract	A common task in data analysis is to find the appropriate data sample whose properties allow us to infer the parameters of the data population. The most frequently dilemma related to sampling is how to determine the optimal *size* of the sample. To solve it, there are typical methods based on asymptotical results from the Central Limit Theorem. However, the effectiveness of such methods is bounded by several considerations as the sampling strategy (simple, stratified, cluster-based, etc.), the size of the population or even the dimensionality of the space of the data. In order to avoid such constraints, we propose a method based on a measure of information of the data in terms of Shannon's Entropy. Our idea is to find the optimal sample of size N whose information is as similar as possible to the information of the population, subject to several constraints. Finding such sample represents a hard optimization problem whose feasible space disallows the use of traditional optimization techniques. To solve it, we resort to Genetic Algorithms. We test our method with synthetic datasets; the results show that our method is suitable. For completeness, we used a dataset from a real problem; the results confirm the effectiveness of our proposal and allow us to visualize different applications.

Keywords (separated by '-')	Sampling data - Genetic algorithms - Shannon's entropy - Feature selection

Finding the Optimal Sample Based on Shannon's Entropy and Genetic Algorithms

Edwin Aldana-Bobadilla[1,2(✉)] and Carlos Alfaro-Pérez[1,2]

[1] Facultad de Ingeniería UNAM, Ciudad Universitaria, México D.F, Mexico
edwynjavier@yahoo.es, carlos.alfaro26@gmail.com
[2] Facultad de Estudios Superiores-UNAM, Naucalpan
Estado de México, Mexico

Abstract. A common task in data analysis is to find the appropriate data sample whose properties allow us to infer the parameters of the data population. The most frequently dilemma related to sampling is how to determine the optimal *size* of the sample. To solve it, there are typical methods based on asymptotical results from the Central Limit Theorem. However, the effectiveness of such methods is bounded by several considerations as the sampling strategy (simple, stratified, cluster-based, etc.), the size of the population or even the dimensionality of the space of the data. In order to avoid such constraints, we propose a method based on a measure of information of the data in terms of Shannon's Entropy. Our idea is to find the optimal sample of size N whose information is as similar as possible to the information of the population, subject to several constraints. Finding such sample represents a hard optimization problem whose feasible space disallows the use of traditional optimization techniques. To solve it, we resort to Genetic Algorithms. We test our method with synthetic datasets; the results show that our method is suitable. For completeness, we used a dataset from a real problem; the results confirm the effectiveness of our proposal and allow us to visualize different applications.

AQ1

Keywords: Sampling data · Genetic algorithms · Shannon's entropy · Feature selection

1 Introduction

The goal of sampling is to choose a representative subset S from a set called population denoted by P. One way in which S may be obtained is by a random process where each element in P has an equal probability of being selected (simple sampling). When this process allows us to choose an element from P more than once, it is called sampling with replacement [1] otherwise it is called sampling without replacement [2]. Alternative ways to obtain S are: systematic sampling [3], stratified sampling [4] and cluster sampling [5]. Regardless of the *sampling strategy*, an important concern is how to determine the cardinality or size of S. Usually this value is determined resorting to asymptotical results from the Central Limit Theorem (CLT) [6]. Assuming a sample S_i of size N drawn from P with mean μ_{Si}, let $\bar{X}$ be a set of means μ_{Si} of the form:

G. Sidorov and S.N. Galicia-Haro (Eds.): MICAI 2015, Part I, LNAI 9413, pp. 1–11, 2015.
DOI: 10.1007/978-3-319-27060-9_29

$$\bar{X} = \{\mu_{S_1}, \mu_{S_2}, \ldots, \mu_{S_M}\} \tag{1}$$

From CLT, it is said that there is a relationship between the mean of $\bar{X}$ denoted as $\mu_{\bar{X}}$ and the mean of the population P denoted as μ which is given by:

$$\mu_{\bar{X}} \cong \mu \tag{2}$$

Likewise, it is said that there is a relationship between the deviation of $\bar{X}$ denoted as $\sigma_{\bar{X}}$, and the standard deviation of the population σ which is given by:

$$\sigma_{\bar{X}} \cong \frac{\sigma}{\sqrt{N}} \tag{3}$$

Since $\sigma_{\bar{X}}$ represents a dispersion measure of the samples S_i (usually called standard error), its value must be as small as possible. There is an optimal value of N that allows satisfying such condition. We illustrate this fact in Fig. 1, with a synthetic dataset in $\mathbb{R}$ of size 6000. Every point is the standard error obtained with different values of N. We can see that the value of N that minimizes the standard error is the closest to the population size. Obviously, for practical purposes, such value is unsuitable. An error value must be assumed to choose a value of N less than the cardinality of P.

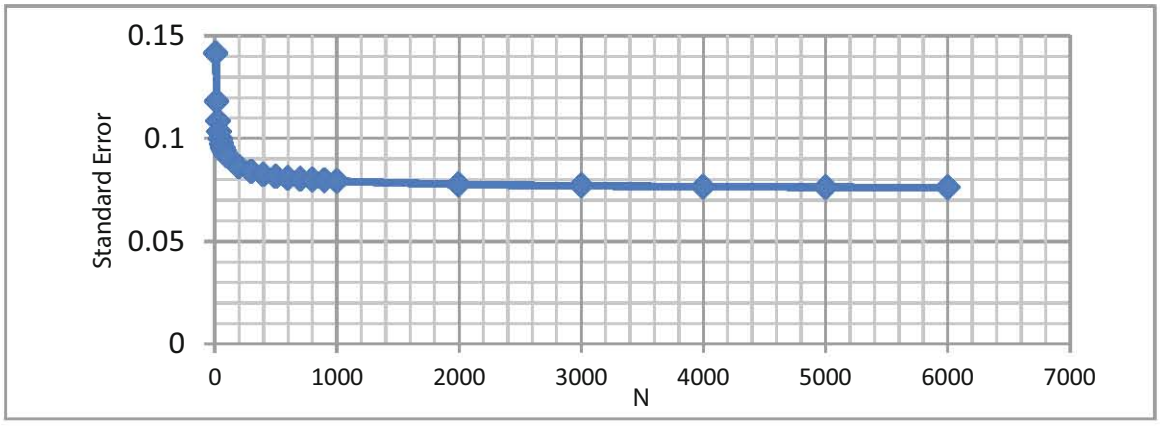

Fig. 1. Standard error in function of the sample size

From (3) and based on such error, the size of a sample is given by:

$$N \cong \frac{\sigma^2}{\sigma_{\bar{X}}^2} \tag{4}$$

From (4), typically the value of $\sigma_{\bar{X}}$ is defined in a discretionary way. From our example (see Fig. 1), if we assume that $\sigma_{\bar{X}} = 0.05$, the value of N will be greater than the size or cardinality of P. It means that not always given a value of $\sigma_{\bar{X}}$ the value of N is appropriate. We are facing an optimization problem which involves to find the breakeven point between N and $\sigma_{\bar{X}}$. An important consideration in the above discussion

is that the asymptotical relationships in (2) and (3) assume that the samples are obtained by a simple random sampling; usually more complex *sampling strategies* are not taken into account. Another consideration is that these asymptotical relationships do not consider random variables in a multidimensional space. We propose a method that allows us to find the optimal sample S' from a population P without the above considerations. The idea is to find a sample of size N whose information is as similar as possible to the information of the population and the value of N is minimal. In order to measure the information, we resort to Shannon's Entropy [7]. The size of the search space is given by:

$$\sum_{N=1}^{|P|} \binom{|P|}{N} = \sum_{N=1}^{|P|} \frac{|P|!}{N!(|P|-N)!} \tag{5}$$

where $\binom{|P|}{N}$ is the number of ways of picking N elements from P. Since this search space is huge, it is necessary to resort to a method that allows us to explore it efficiently. Among the many methods that have arisen, we mention tabu search [8], simulated annealing [9], ant colony optimization [10], particle swarm optimization [11] and evolutionary computation [12]. Furthermore, among the many variations of evolutionary computation, we find evolutionary strategies [13], evolutionary programming [14], genetic programming [15] and genetic algorithms (GAs) [16]. All of these methods are used to find approximate solutions for complex optimization problems. It was proven that an elitist GA always converges to the global optimum [17]. Such a convergence, however, is not bounded in time, and the selection of the GA variation with the best dynamic behavior is very convenient. In this regard, we rely on the conclusions of previous analyses [18, 19], which showed that a breed of GA, called the eclectic genetic algorithm (EGA), achieves the best relative performance.

Having determined the measure of information and chosen the appropriate method to explore the wide search space, in the following section, we present the details of our proposal. The rest of this work has been organized as follows: In Sect. 2 we show the background to guide the discussion about our proposal. We show how to measure the information of the data based on Shannon's Entropy and how to extend such measure to data in multidimensional space. In Sect. 3, we show in detail our proposal. In Sect. 4, we show the experimental methodology and its results. Finally, we present the conclusions and infer several applications.

2 Preliminaries

2.1 Measuring the Expected Information Value

The so-called entropy [20] appeals to an evaluation of the information content of a random variable Y with possible values $\{y_1, y_2, \ldots, y_n\}$. From a statistical viewpoint, the information of the event $(Y = \mathrm{y}_i)$ is inversely proportional to its likelihood. This information is denoted by $I(y_i)$, which can be expressed as:

$$I(y_i) = \log\left(\frac{1}{p(y_i)}\right) = -\log(p(y_i)) \tag{6}$$

From information theory [21], the entropy of Y is the expected value of I. It is given by:

$$H(Y) = -\sum_{i=1}^{n} p(y_i)\log(p(y_i)) \tag{7}$$

Typically, the *log* function may be taken to be log_2, and then, the entropy is expressed in bits; otherwise, as *ln*, in which case the entropy is in nats. We will use log_2 for the computations in this paper. We can visualize P and S as random variables, thus their entropies can be calculated from (7). We want to choose a sample S of **SIZE** N (in what follows denoted as S_N) from P such that:

$$\frac{|H(S_N) - H(P)|}{H(P)} \leq \varepsilon \tag{8}$$

Where ε is a parameter that represents the maximum permissible error between the information of P and S. To calculate the entropy, an important issue is how to determine the probability $p(y_i)$. Since usually the probability distribution function (PDF) of Y is unknown, we approximate such PDF through a method based on quantiles. Such method can be extended to multidimensional random variables ($Y \in \mathbb{R}^n, n \geq 2$). In what follows, we expand the discussion about this.

2.2 Fitting Distribution of Y

We can divide the space of Y into a set of intervals usually called quantiles [22]. The PDF of Y is approximated by the proportion of the elements that lies in each quantile. This is illustrated in Fig. 2.

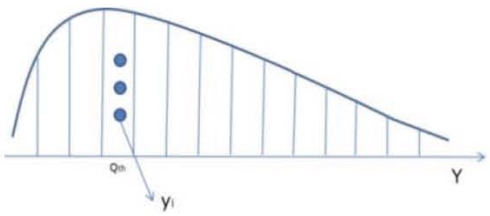

Fig. 2. A possible division of the space of Y in a one-dimensional space

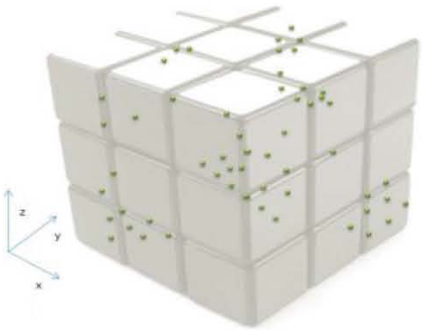

Fig. 3. A possible division of the space of Y in a 3-dimensional space

The above idea may be extended to a multidimensional space, in which case, a quantile is a multidimensional partition of the space of Y as is shown in Fig. 3 ("hyper-quantile"). In both cases the probability $p(y_i)$ is the density of the quantile to which $p(y_i)$ belongs in terms of data contained in it. To determine the number of quantiles in which the space of Y must be divided, typically, Sturges' rule [23] is used. There are other alternative rules which attempt to improve the performance of Sturges's without a normality assumption as Doane's formula [24] and the Rice rule [25]. We prefer the Rice rule, which is to *set the number of intervals to twice the cube root of the number of observations*. In the case of 1000 observations, the Rice rule yields 20 intervals instead of the 11 recommended by Sturges' rule.

Having defined the way to measure the information of P and S_N, in the following section, we present important details to find the optimal S_N.

3 Proposal

3.1 Defining the Objective Function

We want to find the minimal value of N that allows us to obtain a sample S_N from P, whose entropy is as close as possible to entropy of P. Finding such sample is an optimization problem of the form:

$$\begin{gathered} \textit{Minimize}: \ f(N) = \frac{N}{|P|} \\ \textit{subject to}: \\ \frac{|H(S_N) - H(P)|}{H(P)} \leq \varepsilon \\ N \in [2, |P|) \end{gathered} \tag{9}$$

The value of the objective function tends to 1 when the value of N is close to $|P|$. Otherwise this value tends to decrease as the value of N is away from $|P|$. As mentioned, ε is a parameter that represents the maximum permissible error between the information of P and S. The problem in (9) is a constrained optimization problem. To solve it, we use EGA with a constraint handling strategy which is described in Subsect. 3.3.

3.2 Encoding the Problem

EGA proposes M candidate values of N which allow us to obtain M samples S_N. The fitness value of these candidates is determined by the objective function. Additionally each candidate must satisfy the constraints defined in (9). The fitness of those candidates that does not satisfy such constraints is punished through a penalty function (see Subsect. 3.3). Each candidate is encoded as a binary string of length 32. EGA generates a population of candidates (binary strings) and evaluates them in accordance with their **integer** representation and the objective function (9). Evolution takes place after the repeated application of the genetic operators [18, 19]. *It is important to remark that our*

method is independent of the sampling strategy to select the elements from P. We can choose any strategy and use it throughout of the evolutionary process of EGA.

3.3 Constraints Handling Strategy

To solve a constrained optimization problem, the most common way is resorting to a penalty function [26]. In this approach, the objective function in (9) can be transformed as follows:

$$F(N) = \begin{cases} f(N) & \text{if } N \text{ is a feasible solution} \\ f(N) + penalty(N) & \text{otherwise} \end{cases} \tag{10}$$

There are many variations of penalty functions. Based on the results of a comprehensive analysis reported in [27], we use the method that exhibited the best performance, where the objective function $f(N)$ is transformed as follows:

$$F(N) = \begin{cases} \left[K - \sum_{i=1}^{s} \frac{K}{p}\right] & s \neq p \\ f(N) & \text{otherwise} \end{cases} \tag{11}$$

where K is a large constant $[\mathrm{O}(10^9)]$, p is the number of constraints and s is the number of these which have been satisfied.

3.4 Getting the Optimal Sample

When the evolutionary process of EGA is finished, the population will have the best candidate whose genome (the value of N) allows us to find the optimal sample S_N. Given a population[1] P (dataset), such process involves the following:

1. Generate a random set C with candidate solutions in accordance to the problem encoding.
2. For each candidate or individual in C, decode its genome in order to obtain a value of N and a sample S_N from P.
3. Determine the fitness value of each individual based on objective function (9).
4. Determine the feasibility of each individual in C and apply the penalty function (if necessary).
5. Sort C in ascending order, based on the fitness values.
6. Apply genetic operators of EGA (see [18, 19]).
7. Repeated 2-4 until convergence criteria are met (usually the number of generations).
8. Select the top candidate from C; decode it in order to obtain a value of N.
9. Obtain a sample S_N from P. This is the optimal sample which satisfies: $\frac{|H(S_N) - H(P)|}{H(P)} \leq \varepsilon$

[1] To avoid ambiguities, the word *population* and the term P refer to the dataset to be sampled rather than the set of candidate solutions of EGA. Instead, this last set is denoted as C.

As mentioned, we can choose *any sampling strategy* to obtain S_N throughout of the evolutionary process. The value of ε must be given a priori, we set $\varepsilon = 0.01$, which represents an error of 1 % between the entropies of the sample and the population.

3.5 Setting Additional Parameters

The EGA was executed with the following parameter values: $Pc = 0.90$, $Pm = 0.009$, $|C| = 70$, $G = 300$. It is based on a mentioned study [18], which showed that from a statistical view point, EGA converges to the optimal solution around such values when the problems are demanding (those with a non-convex feasible space).

4 Results

We wanted to show some **preliminary results** as evidence of the effectiveness of our method. Subsequently, we defined a performance measure relative to the ratio between the size of S_N and P. We executed systematically a set of experiments (with synthetic datasets) which allowed us to find the behavior of such ratio from the statistical view point. Finally, we show the effectiveness of our method with a real problem.

4.1 Preliminary Results

We executed preliminary experiments whose results allow us to show that our method is promissory. The datasets in such experiments included:

- A dataset of 10000 elements in a one-dimensional space drawn from a gaussian distribution (see Fig. 4).
- A dataset of 10000 elements in a bi-dimensional space drawn from a gaussian distribution (see Fig. 5).
- A dataset of 10000 elements in bi-dimensional space that represent a sinusoidal function (see Fig. 6).

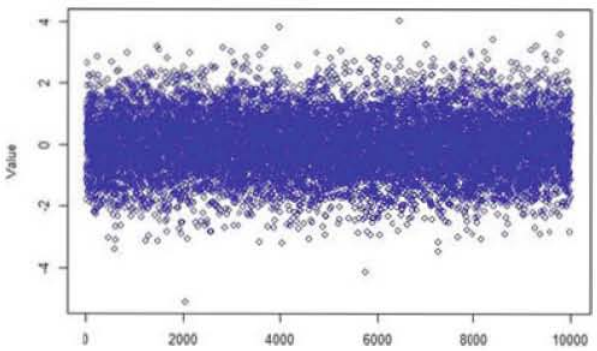

Fig. 4. Gaussian dataset in $\mathbb{R}$

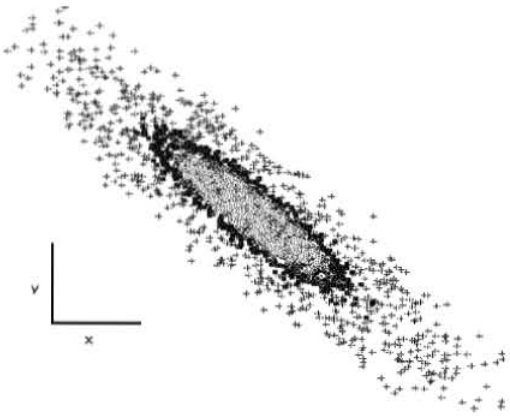

Fig. 5. Gaussian dataset in $\mathbb{R}^2$

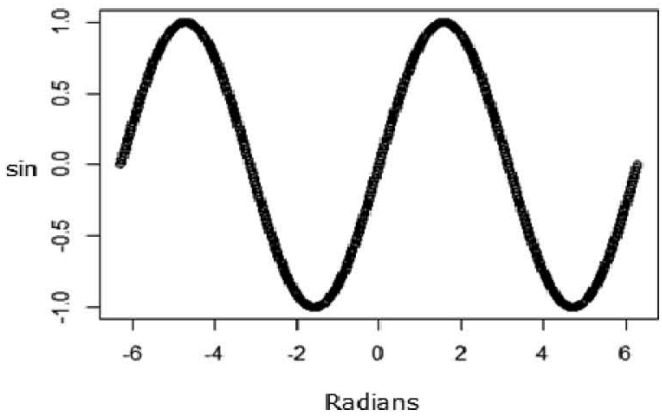

Fig. 6. Sinusoidal dataset

Our hypothesis is that our method will find the optimal sample (one with the minimal value of N) from these datasets, retaining their properties (e.g. probability distribution or spatial arrangement). In Figs. 7, 8, 9, 10, 11 and 12 we can see that the samples retain several properties of the population. Additional properties are shown in Table 1. As extended work, we will show that in general, the properties of the PDF (e.g. unimodality, skweness, kurtosis, etc.) are similar to those of the dataset. In this regard the reader can find an interesting work in [28].

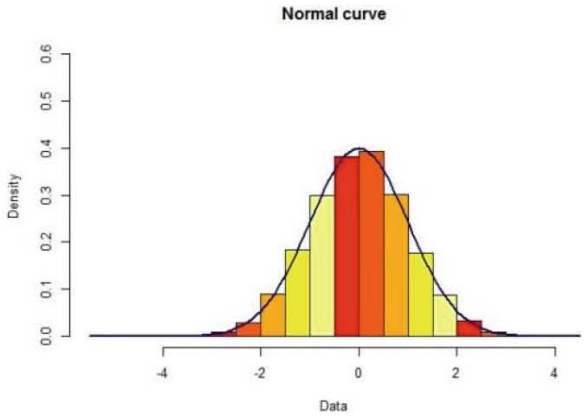

Fig. 7. Density of the dataset in $\mathbb{R}$

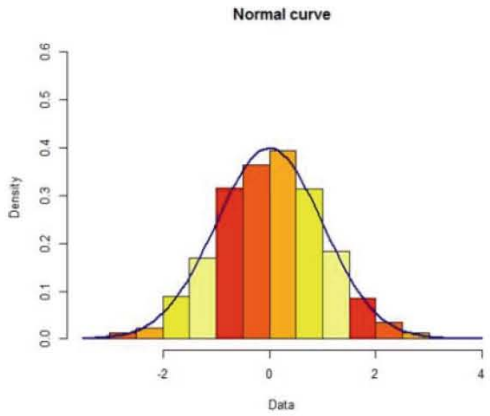

Fig. 8. Density of the sample in $\mathbb{R}$

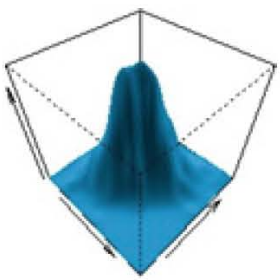

Fig. 9. Density of the dataset in $\mathbb{R}^2$

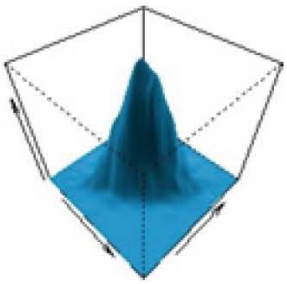

Fig. 10. Density of the sample in $\mathbb{R}^2$

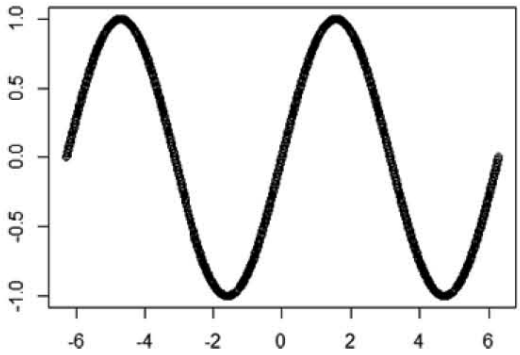

Fig. 11. Spatial arrangement of the sinusoidal dataset

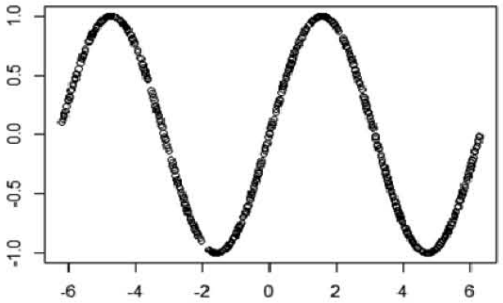

Fig. 12. Spatial arrangement of the sinusoidal sample

Table 1. Additionaly properties of the datasets and the samples

	Dataset 1		Dataset 2		Dataset 3	
	Population	Sample	Population	Sample	Population	Sample
Size	10000	1860	10000	1512	10000	768
Entropy	3.0534	3.0527	3.9261	3.9937	3.7281	3.7194

4.2 Measuring the Performance

To measure the performance of our method, we resort to the *space saving metric* (SS) from compression data which is given by:

$$SS = 1 - \frac{N}{|P|} \tag{12}$$

A large value of SS (closer to 1) implies better performance. We calculated such metric with a wide set of experiments (about 5000) which included random datasets of size 1000, 5000, 10000 and 100000. The results are shown in Table 2. For completeness, we determine the confidence interval of the results with a p-value of 0.05.

Table 2. Summary of results obtained with differents datasets (gaussian a non-gaussian)

SS	0.7895
Standard deviation	0.3181
Lower limit	0.7806
Upper limit	0.7983
Confidence level	95 %

The experiments shows that in average the sampling process allows us to reduce the size of the dataset in more than 70 %. In this sense our method can be considered a lossy compression method.

4.3 Real World Dataset

We tested our method with a dataset whose elements represent information of the handwritten digit recognition problem. The problem is to separate the highly confusible digits '4' and '9'. Such dataset was obtained from UCI repository [29]. This dataset has 6000 instances (elements) with 5000 attributes. We applied our method in order to *reduce the number of such attributes*. For each instance, we executed our method to obtain the most informative attributes (those whose entropy is close to entropy of all attributes of the instance). Based on the above, 6000 experiments were executed. These experiments allow us to find the relative frequency of each attribute. We chose those with frequency greater than 0.15. Such decision allows us to reduce the number of attributes to 890. To test that such reduction retains the properties of the original data, we applied a Multi-layer Perceptron Network (MLPN) with the reduced dataset. The labeling set obtained by MLPN was compared with the labeling set of the original dataset. The proportion of identical elements (labels) between these sets was 97.68 %.

5 Conclusions

A new sampling method based on the entropy has been defined in order to find a sample of minimal size from a dataset (population). Finding the optimal sample involves an optimization problem that requires an efficient method to explore the huge feasible space. We use EGA as the best alternative. A first approach allows us to determine that our method is able to find a sample from a dataset that retains the original properties related to the probability distribution or spatial arrangement. Based on these results a wide set of experiments on synthetic datasets was executed. On average the method achieved a value of *SS* of 0.78. It means that we can reduce the size of a dataset in more than 70 % and obtain the optimal samples that retain the information of the original datasets with an error of 1 % (given by the value of ε). Based on the above, we can tackle many applications that require lossy compression data. We showed that our method can be applied to those problems related to feature selection, where removing redundant attributes is compulsory.

References

1. Mukhopadhyay, P.: Theory and Methods of Survey Sampling. PHI Learning Pvt. Ltd., New Delhi (2009)
2. Sukhatme, P.V.: Sampling Theory of Surveys with Applications. Iowa State University Press, Ames (1957)
3. Israel, G.D.: Sampling the evidence of extension program impact. University of Florida Cooperative Extension Service, Institute of Food and Agriculture Sciences, EDIS (1992)
4. Cochran, W.G.: Sampling Techniques. Wiley, New York (2007)
5. Särndal, C.-E., Swensson, B., Wretman, J.: Model Assisted Survey Sampling. Springer, New York (2003)

6. Barany, I., Vu, V.: Central limit theorems for Gaussian polytopes. Ann. Probab. **34**, 1593–1621 (2007)
7. Shannon, C.E.: A mathematical theory of communication. ACM SIGMOBILE Mob. Comput. Commun. Rev. **5**(1), 3–55 (2001)
8. Glover, F.: Tabu search-part I. ORSA J. Comput. **1**(3), 190–206 (1989)
9. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)
10. Dorigo, M., Birattari, M.: Ant colony optimization. In: Sammut, C., Webb, G.I. (eds.) Encyclopedia of Machine Learning, pp. 36–39. Springer, New York (2010)
11. Kennedy, J.: Particle Swarm Optimization. Encyclopedia of Machine Learning, pp. 760–766. Springer, New York (2010)
12. Spears, W.M., et al.: An overview of evolutionary computation. In: Brazdil, P.B. (ed.) ECML 1993. LNCS, vol. 667, pp. 442–459. Springer, Heidelberg (1993)
13. Geritz, S.A.H., Mesze, G., Metz, J.A.J.: Evolutionarily singular strategies and the adaptive growth and branching of the evolutionary tree. Evol. Ecol. **12**(1), 35–57 (1998)
14. Kim, J.-H., Myung, H.: Evolutionary programming techniques for constrained optimization problems. IEEE Trans. Evol. Comput. **1**(2), 129–140 (1997)
15. Koza, J.R., Bennett III, F.H., Stiffelman, O.: Genetic programming as a Darwinian invention machine. In: Langdon, W.B., Fogarty, T.C., Nordin, P., Poli, R. (eds.) EuroGP 1999. LNCS, vol. 1598, pp. 93–108. Springer, Heidelberg (1999)
16. Goldberg, D.E., Holland, J.H.: Genetic algorithms and machine learning. Mach. Learn. **3**(2), 95–99 (1988)
17. Rudolph, G.: Convergence analysis of canonical genetic algorithms. IEEE Trans. Neural Netw. **5**(1), 96–101 (1994)
18. Kuri-Morales, A., Aldana-Bobadilla, E.: The best genetic algorithm I. In: Castro, F., Gelbukh, A., González, M. (eds.) MICAI 2013, Part II. LNCS, vol. 8266, pp. 1–15. Springer, Heidelberg (2013)
19. Morales, A.K., Quezada, C.V.: A universal eclectic genetic algorithm for constrained optimization. Proceedings of the 6th European Congress on Intelligent Techniques and Soft Computing, vol. 1 (1998)
20. Shannon, C.E.: A note on the concept of entropy. Bell Syst. Tech. J. **27**, 379–423 (1948)
21. Shannon, C.E., Weaver, W.: The mathematical theory of information (1949)
22. Hyndman, R.J., Fan, Y.: Sample quantiles in statistical packages. Am. Stat. **50**(4), 361–365 (1996)
23. Hyndman, R.J.: The problem with Sturges' rule for constructing histograms. Monash University (1995)
24. Doane, D.P.: Aesthetic frequency classifications. Am. Stat. **30**(4), 181–183 (1976)
25. Soo, N.H., Halim, Y.: Feature selection methodology in quality data mining (2004)
26. White, D.J., Anandalingam, G.: A penalty function approach for solving bi-level linear programs. J. Glob. Optim. **3**(4), 397–419 (1993)
27. Kuri-Morales, Á.F., Gutiérrez-García, J.O.: Penalty function methods for constrained optimization with genetic algorithms: a statistical analysis. In: Coello Coello, C.A., de Albornoz, Á., Sucar, L., Battistutti, O.C. (eds.) MICAI 2002. LNCS (LNAI), vol. 2313, pp. 108–117. Springer, Heidelberg (2002)
28. Kuri-Morales, A., Rodríguez-Erazo, F.: A search space reduction methodology for data mining in large databases. Eng. Appl. Artif. Intell. **22**(1), 57–65 (2009)
29. Lichman, M.: UCI Machine Learning Repository. http://archive.ics.uci.edu/ml/datasets/Gisette. University of California, School of Information and Computer Science, Irvine (2013)

Author Query Form

Book ID :	**393961_1_En**
Chapter No.:	**29**

Springer
the language of science

Please ensure you fill out your response to the queries raised below and return this form along with your corrections

Dear Author
During the process of typesetting your chapter, the following queries have arisen. Please check your typeset proof carefully against the queries listed below and mark the necessary changes either directly on the proof/online grid or in the 'Author's response' area provided below

Query Refs.	Details Required	Author's Response
AQ1	Please check and confirm if the authors and their respective affiliations have been correctly identified. Amend if necessary.	

MARKED PROOF

Please correct and return this set

Please use the proof correction marks shown below for all alterations and corrections. If you wish to return your proof by fax you should ensure that all amendments are written clearly in dark ink and are made well within the page margins.

Instruction to printer	*Textual mark*	*Marginal mark*
Leave unchanged	under matter to remain	
Insert in text the matter indicated in the margin		New matter followed by or
Delete	through single character, rule or underline or through all characters to be deleted	or
Substitute character or substitute part of one or more word(s)	through letter or through characters	new character or new characters
Change to italics	under matter to be changed	
Change to capitals	under matter to be changed	
Change to small capitals	under matter to be changed	
Change to bold type	under matter to be changed	
Change to bold italic	under matter to be changed	
Change to lower case	Encircle matter to be changed	
Change italic to upright type	(As above)	
Change bold to non-bold type	(As above)	
Insert 'superior' character	through character or where required	or under character e.g. or
Insert 'inferior' character	(As above)	over character e.g.
Insert full stop	(As above)	
Insert comma	(As above)	
Insert single quotation marks	(As above)	or and/or or
Insert double quotation marks	(As above)	or and/or or
Insert hyphen	(As above)	
Start new paragraph		
No new paragraph		
Transpose		
Close up	linking characters	
Insert or substitute space between characters or words	through character or where required	
Reduce space between characters or words	between characters or words affected	

Referencias

[1] S. Chaudhuri and U. Dayal, "An overview of data warehousing and olap technology," *ACM Sigmod record*, vol. 26, no. 1, pp. 65–74, 1997.

[2] J. Han and M. Kamber, *Data Mining, Southeast Asia Edition: Concepts and Techniques*. Morgan kaufmann, 2006.

[3] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery in databases," *AI magazine*, vol. 17, no. 3, p. 37, 1996.

[4] D. J. Hand, H. Mannila, and P. Smyth, *Principles of data mining*. MIT press, 2001.

[5] X.-K. Song, *Correlated data analysis: modeling, analytics, and applications*. Springer, 2007.

[6] J. Fan and R. Li, "Statistical challenges with high dimensionality: Feature selection in knowledge discovery," *arXiv preprint math/0602133*, 2006.

[7] A. Agresti, *An introduction to categorical data analysis*, vol. 135. Wiley New York, 1996.

[8] S. E. Fienberg, *The analysis of cross-classified categorical data*. Springer, 2007.

[9] G. E. Barton, R. C. Berwick, and E. S. Ristad, *Computational complexity and natural language*. MIT press, 1987.

[10] C. H. Papadimitriou, *Computational complexity*. John Wiley and Sons Ltd., 2003.

[11] M. Krzywinski and N. Altman, "Points of significance: Significance, p values and t-tests," *Nature methods*, vol. 10, no. 11, pp. 1041–1042, 2013.

[12] D. Yates, D. Moore, and D. S. Starnes, *The practice of statistics: TI-83/89 graphing calculator enhanced*. Macmillan, 2002.

[13] K. Black, *Business statistics: for contemporary decision making*. John Wiley & Sons, 2011.

[14] C.-E. Särndal, B. Swensson, and J. Wretman, *Model assisted survey sampling*. Springer, 2003.

[15] G. D. Israel, *Sampling the evidence of extension program impact*. University of Florida Cooperative Extension Service, Institute of Food and Agriculture Sciences, EDIS, 1992.

[16] M. R. Spiegel, "Estadística. segunda edición. serie schaum," 1991.

[17] A. W. Van der Vaart, *Asymptotic statistics*, vol. 3. Cambridge university press, 2000.

[18] T.-J. Yao, C. B. Begg, and P. O. Livingston, "Optimal sample size for a series of pilot trials of new agents," *Biometrics*, pp. 992–1001, 1996.

[19] P. Müller, G. Parmigiani, C. Robert, and J. Rousseau, "Optimal sample size for multiple testing: the case of gene expression microarrays," *Journal of the American Statistical Association*, vol. 99, no. 468, pp. 990–1001, 2004.

[20] S. Walter, M. Eliasziw, and A. Donner, "Sample size and optimal designs for reliability studies," *Statistics in medicine*, vol. 17, no. 1, pp. 101–110, 1998.

[21] H. Liu and H. Motoda, *Instance selection and construction for data mining*, vol. 608. Springer Science & Business Media, 2013.

[22] K. C. Gowda and G. Krishna, "The condensed nearest neighbor rule using the concept of mutual nearest neighborhood," *IEEE Transactions on Information Theory*, vol. 25, no. 4, pp. 488–490, 1979.

[23] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," *Systems, Man and Cybernetics, IEEE Transactions on*, no. 3, pp. 408–421, 1972.

[24] D. G. Lowe, "Similarity metric learning for a variable-kernel classifier," *Neural computation*, vol. 7, no. 1, pp. 72–85, 1995.

[25] D. Kibler and D. Aha, "Learning representative exemplars of concepts: An initial case study," in *Proc. of the 4th International Workshop on Machine Learning*, pp. 24–30, 1987.

[26] D. R. Wilson and T. R. Martinez, "Reduction techniques for instance-based learning algorithms," *Machine learning*, vol. 38, no. 3, pp. 257–286, 2000.

[27] K. Kalegele, H. Takahashi, J. Sveholm, K. Sasai, G. Kitagata, and T. Kinoshita, "On-demand data numerosity reduction for learning artifacts," in *Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on*, pp. 152–159, IEEE, 2012.

[28] D. B. Skalak, "Prototype and feature selection by sampling and random mutation hill climbing algorithms," in *Proceedings of the eleventh international conference on machine learning*, pp. 293–301, 1994.

[29] C. R. Reeves and D. R. Bush, "Using genetic algorithms for training data selection in rbf networks," in *Instance selection and construction for data mining*, pp. 339–356, Springer, 2001.

[30] J. R. Cano, F. Herrera, and M. Lozano, "Using evolutionary algorithms as instance selection for data reduction in kdd: an experimental study," *Evolutionary Computation, IEEE Transactions on*, vol. 7, no. 6, pp. 561–575, 2003.

[31] C. E. Shannon and W. Weaver, "The mathematical theory of information," 1949.

[32] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.

[33] H. Oja, *Multivariate nonparametric methods with R: an approach based on spatial signs and ranks*. Springer Science & Business Media, 2010.

[34] R. J. Hyndman and Y. Fan, "Sample quantiles in statistical packages," *The American Statistician*, vol. 50, no. 4, pp. 361–365, 1996.

[35] R. J. Hyndman, "The problem with sturges rule for constructing histograms," *Monash University*, 1995.

[36] D. P. Doane, "Aesthetic frequency classifications," *The American Statistician*, vol. 30, no. 4, pp. 181–183, 1976.

[37] D. M. Lane, *Online statistics education: An interactive multimedia course of study*. 2015.

[38] J. E. Beasley and J. E. Beasley, *Advances in linear and integer programming*. Clarendon Press Oxford, 1996.

[39] G. L. Nemhauser and L. A. Wolsey, *Integer and combinatorial optimization*, vol. 18. Wiley New York, 1988.

[40] T. Odzijewicz and D. F. Torres, "Calculus of variations with classical and fractional derivatives," *arXiv preprint arXiv:1007.0567*, 2010.

[41] K. Y. Lee and M. A. El-Sharkawi, *Modern heuristic optimization techniques: theory and applications to power systems*, vol. 39. John Wiley & Sons, 2008.

[42] F. Glover, "Tabu search-part i," *ORSA Journal on computing*, vol. 1, no. 3, pp. 190–206, 1989.

[43] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, *et al.*, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[44] M. Dorigo and M. Birattari, "Ant colony optimization," in *Encyclopedia of machine learning*, pp. 36–39, Springer, 2010.

[45] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*, pp. 760–766, Springer, 2010.

[46] W. M. Spears, K. A. De Jong, T. Bäck, D. B. Fogel, and H. De Garis, "An overview of evolutionary computation," in *Machine Learning: ECML-93*, pp. 442–459, Springer, 1993.

[47] S. A. Geritz, G. Mesze, J. A. Metz, *et al.*, "Evolutionarily singular strategies and the adaptive growth and branching of the evolutionary tree," *Evolutionary ecology*, vol. 12, no. 1, pp. 35–57, 1998.

[48] J.-H. Kim and H. Myung, "Evolutionary programming techniques for constrained optimization problems," *Evolutionary Computation, IEEE Transactions on*, vol. 1, no. 2, pp. 129–140, 1997.

[49] J. R. Koza, F. H. Bennett III, and O. Stiffelman, "Genetic programming as a darwinian invention machine," in *Genetic Programming*, pp. 93–108, Springer, 1999.

[50] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Machine learning*, vol. 3, no. 2, pp. 95–99, 1988.

[51] G. Rudolph, *Convergence properties of evolutionary algorithms*. Kovac, 1997.

[52] A. Kuri-Morales and E. Aldana-Bobadilla, "The best genetic algorithm i," in *Advances in Soft Computing and Its Applications*, pp. 1–15, Springer, 2013.

[53] A. K. Morales and C. V. Quezada, "A universal eclectic genetic algorithm for constrained optimization," in *Proceedings of the 6th European congress on intelligent techniques and soft computing*, vol. 1, pp. 518–522, Citeseer, 1998.

[54] D. J. White and G. Anandalingam, "A penalty function approach for solving bi-level linear programs," *Journal of Global Optimization*, vol. 3, no. 4, pp. 397–419, 1993.

[55] A. F. Kuri-Morales and J. Gutiérrez-García, "Penalty function methods for constrained optimization with genetic algorithms: A statistical analysis," in *MICAI 2002: Advances in Artificial Intelligence*, pp. 108–117, Springer, 2002.

[56] A. F. Kuri-Morales, E. Aldana-Bobadilla, and I. López-Peña, "The best genetic algorithm ii," in *Advances in Soft Computing and Its Applications*, pp. 16–29, Springer, 2013.

[57] M. Lichman, "Uci machine learning repository, yeast dataset." http://archive.ics.uci.edu/ml/datasets/Abalone, 1996. Accessed: 2015-03-22.

[58] M. Lichman, "Uci machine learning repository, yeast dataset." http://archive.ics.uci.edu/ml/datasets/Car+Evaluation, 1996. Accessed: 2015-03-22.

[59] M. Lichman, "Uci machine learning repository, yeast dataset." http:

//archive.ics.uci.edu/ml/datasets/Census+Income, 1996. Accessed: 2015-03-22.

[60] M. Lichman, "Uci machine learning repository, yeast dataset." http://archive.ics.uci.edu/ml/datasets/Hepatitis, 1996. Accessed: 2015-03-22.

[61] M. Lichman, "Uci machine learning repository, yeast dataset." http://archive.ics.uci.edu/ml/datasets/Yeast, 1996. Accessed: 2015-03-22.

[62] A. Agresti, *Analysis of ordinal categorical data*, vol. 656. John Wiley & Sons, 2010.

[63] L. F. Shampine, R. C. Allen, and S. Pruess, *Fundamentals of numerical computing*. Wiley New York, 1997.

[64] M. Lichman, "UCI machine learning repository," 2013.

[65] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT Press, 1992.