Generalización del uso de prototipos para el diseño de redes neuronales tipo MLP con una capa oculta

Adolfo Díaz Sardiñas, Rafael Bello Pérez *

Resumen

El algoritmo Backpropagation (BP) ha causado un profundo impacto en las aplicaciones de redes neuronales de tipo Multilayer Perceptrons (MLP) mostrando sus potencialidades en la obtención de buenas soluciones para muchos problemas. Sin embargo, posee importantes limitaciones: necesidad de gran tiempo de entrenamiento y sensibilidad ante la presencia de mínimos locales. Otro problema es la topología de la red; la determinación del número exacto de capas y neuronas ocultas en muchas ocasiones no resulta una tarea fácil. En este trabajo abordamos, para redes MLP con una capa oculta, el problema de la obtención de un buen conjunto inicial de pesos, la determinación del número de neuronas ocultas y algunas consideraciones sobre la generación de los prototipos. Se demuestra que para este tipo de redes es posible realizar un preprocesamiento de los patrones de entrenamiento que posibilita la obtención de la arquitectura óptima y el conjunto inicial de pesos que le permiten a BP aumentar su eficiencia. Se presenta un algoritmo para la generación de la capa oculta de la red y la determinación del conjunto inicial de pesos, a partir de la información aportada por los prototipos o representantes de clases. Finalmente, se presentan los resultados donde se muestra que la inicialización de redes BP con prototipos generalmente resulta en 1) una reducción drástica del tiempo de entrenamiento, 2) aumento de su robustez contra la posibilidad de quedar atrapado en mínimos locales y 3) mejor generalización.

---------- *Palabras clave:* inicialización, redes neuronales hacia adelante, retropropagación del error, prototipos, aprendizaje supervisado, reconocimiento de patrones.

Abstract

This paper addresses the problem of initializing the topology and weights in multilayer networks with one hidden layer. The presented method use reference patterns or prototypes to select the number of hidden units and the initial weight set, resulting on a network very close to a global minima. This scheme applies to pattern recognition tasks, as well as to the approximation of continuous functions. Issues related to the processing of input patterns and to the generation of prototypes are discussed. Finally, simulation results are presented, showing that initializing backpropagation networks with prototypes generally results in a drastic reduction in training time.

---------- *Key words:* initialization, feedforward neural networks, backpropagation, prototypes, supervised learning, pattern recognition.

* Facultad de Matemática, Física y Computación, Universidad Central de Las Villas, Cuba. E-mail: rbellop@uclv.etecsa.cu.

1. Introducción

Aunque los mecanismos de aprendizaje para redes multicapas se conocen desde hace más de treinta años, la ausencia de algoritmos de entrenamiento adecuados ha conspirado contra su uso exitoso en aplicaciones prácticas [1]. Investigaciones relativamente recientes en métodos de entrenamiento supervisados para redes de este tipo, utilizando para cada capa una función de discriminación lineal, han provocado la extensión y éxito que actualmente presenta. Las redes multicapas (MLP) pueden implementar mapas entrada/salida o separaciones de clases en regiones arbitrariamente complejas. El atributo más importante es que puede aprender a mapear cualquier complejidad. Su aprendizaje es basado en la repetición de presentaciones de los ejemplos de entrenamiento y ya entrenada puede producir resultados sorprendentes y generalizaciones en aplicaciones donde las derivaciones explícitas de mapeos y el descubrimiento de relaciones es casi imposible por otra vía.

Algunos de los aspectos fundamentales que frecuentemente se ponen de manifiesto en las aplicaciones de redes neuronales de tipo MLP que utilizan Backpropagation como algoritmo de aprendizaje [2, 3, 4] son "¿Qué tamaño debe tener la red para realizar la tarea deseada? ¿Cuál es la cantidad de capas y neuronas ocultas que se debe emplear? o ¿Cuál es el conjunto inicial de pesos que debe utilizarse? Las respuestas a estas cuestiones están directamente relacionadas con la capacidad de la red y pueden ser dadas independientemente del algoritmo de aprendizaje empleado.

Ha sido demostrado que una red tipo MLP con cuatro capas, dos de ellas ocultas, puede resolver problemas de clasificación arbitrarios. Irie y Miyake [5] probaron que una red con tres capas, una de ellas oculta, usando BP y un número infinito de nodos en la capa oculta, también puede resolver problemas arbitrarios. Sin embargo este resultado posee poco valor práctico ya que no puede ser utilizado para determinar el tamaño (número de nodos) de una capa oculta en aplicaciones reales. Para ayudar a este proceso también se ha mostrado que los nodos de la capa oculta corresponden a regiones de decisión separadas, en las cuales los ejemplos de entrenamiento son separados. Kung y Hwang [6] usaron aplicaciones de proyecciones algebraicas para especificar cuántos de estos nodos deben ser creados. Estas aplicaciones, sin embargo, se basan en el conocimiento de propiedades de los datos de entrenamiento, las cuales son difíciles de obtener en aplicaciones reales. Por tanto, la determinación de la arquitectura para redes tipo MLP, permanece como un aspecto importante de investigación.

Por otro lado para que BP converja es extremadamente importante comenzar con un buen conjunto de pesos [7], el cual usualmente es determinado de forma aleatoria. Si el entrenamiento no es satisfactorio, sea por la falta de convergencia o porque no hubo generalización, entonces es necesario seleccionar otro conjunto inicial de pesos y volver a entrenar, lo que ocasiona una gran pérdida de tiempo.

En este trabajo nuestro objetivo es formalizar un algoritmo que permita generar redes neuronales de tipo MLP a partir de la información que se puede obtener al procesar el conjunto de ejemplos de entrenamiento con cualquiera de los métodos conocidos de clusterización. Como antecedente tenemos los modelos propuestos por Weymaere y Martens [8], Smyth [9] y Denoeux y Lengellé [10]; cada uno de los cuales están referidos a condiciones y métodos de clusterización muy específicos.

2. Descripción del método

Considere un problema de clasificación que consista en la asignación de vectores de $\Re^n$ a C clases predefinidas. Sea P el conjunto de K vectores de referencia con clasificación conocida. Los prototipos pueden ser todo el conjunto de datos de entrenamiento o algunos extraídos según un determinado procedimiento de síntesis que permita obtener representantes típicos para una de las clases C. Una vía común para usar estos proto-

tipos o vectores de referencia para obtener una regla de decisión es clasificar cada patrón x en la clase que su representante o prototipo esté más cercano a x de acuerdo con una métrica determinada. Un vector de entrada que esté muy lejos de todos o cercano a varios prototipos de diferentes clases, puede ser descartado opcionalmente.

Mediante los algoritmos conocidos de clusterización y agrupamiento es posible obtener descripción confiable de los datos de entrada en términos de un limitado número de clusters (= conjunto de observaciones), cada uno representado por un centro de gravedad y un conjunto de desviaciones estándar. Estos sistemas son rápidos de entrenar y por lo general este proceso se basa en el agrupamiento de datos parecidos en clases, usualmente asumiendo alguna forma paramétrica para la distribución e ignorando la información entre las clases.

Consideremos dos centros de clusters P y Q. Podemos obtener una ecuación que determine una frontera entre ellos de la siguiente forma:

$$X^T(P - Q) - \frac{1}{2}\left(P^T P - Q^T Q\right) = 0 \quad (1)$$

donde X es cualquier punto en uno de los lados del hiperplano determinado por (1). Definimos entonces la función:

$$f(X) = X^T(P - Q) - \frac{1}{2}\left(P^T P - Q^T Q\right) \quad (2)$$

cuyo signo determinará en qué lado de la frontera se encuentra X y la magnitud es la distancia que lo separa del hiperplano. Por otra parte, en un MLP de una capa oculta, los nodos de dicha capa forman hiperplanos de separación en el espacio de rasgos, lo que nos permite establecer un método para la determinación de los nodos ocultos a partir de las fronteras que se forman con los centros de clusters obtenidos por cualquier algoritmo de clusterización. Por supuesto que muchos de ellos redundantes. Con n centros de clusters podemos tener $n \times (n - 1)/2$ posibles fronteras entre ellos y correspondientes a potenciales nodos ocultos. Para la selección de las fronteras más útiles y su utilización como nodos ocultos de un MLP, Smyth [9], propone el siguiente método:

1. Formar una lista (lista de fronteras) de todas las funciones que separan los clusters (ignorando aquellas entre clusters de la misma clase), ordenada por la distancia.
2. Mientras la lista de fronteras no esté vacía:
   a. Tomar como hiperplano actual el que esté en el tope de la pila e insertarlo en la lista de nodos ocultos.
   b. El resto de las fronteras en la lista de fronteras cuyas clases quedan "bien separadas" por el hiperplano actual son eliminadas de la lista de fronteras.

"Bien separadas" significa que sus centros de clusters queden separados por una magnitud mayor o igual a un valor predefinido. El efecto de este proceso es que se seleccionan los hiperplanos que separan los clusters más cercanos y que separan otros clusters como efecto colateral. Como se ha señalado no se tienen en cuenta los hiperplanos que separan clusters de la misma clase debido a que el proceso de clusterización se puede realizar de dos formas: global (un cluster por clase) o por clase (varios clusters por clase); esta segunda variante es preferible ya que una descripción característica por clases diferentes es más discriminativa que una descripción global de los datos.

Hasta aquí contamos con un método que nos permite seleccionar de forma óptima los nodos de la capa oculta; faltaría establecer la forma de calcular los pesos de la capa de entrada a la oculta y determinar los valores de los umbrales para dicha capa.

$$w_i = \delta(p_i - q_i) \quad (3)$$

donde W_i es el peso de la entrada i al nodo que separa los centros de cluster $P(= (p_1, \ldots, p_m))$ y Q $(= (q_1, \ldots, q_m))$. El umbral estaría dado por:

$$b = -\frac{\delta}{2}\sum_{i=1}^{m}\left(p_i^2 - q_i^2\right) \quad (4)$$

siendo δ una constante de escalada arbitraria y m el número de rasgos.

Esta forma de calcular los pesos de la capa de entrada a la oculta y los umbrales para los nodos ocultos nos permite que estos funcionen en forma similar a (2).

Lo que resta entonces, es estimar los pesos de la capa oculta a la capa de salida. Para cada patrón de entrenamiento (centros de clusters), la activación de la capa oculta puede ser calculada ya que tanto las entradas como los pesos de la capa de entrada a la oculta son conocidos; además se conoce la salida deseada para la capa de salida. La salida deseada para la clasificación es un vector con el nodo de salida correspondiente a la clase a que pertenece el patrón con el mayor nivel de activación: esto resulta en un conjunto de inecuaciones muy difíciles de satisfacer.

Ya que los valores de activación son funciones fijas de la suma pesada de los nodos de entrada, la función sigmoidal puede ser invertida para obtener los valores presigmoidales de los nodos de salida. El problema ahora es reducir las ecuaciones a un conjunto de ecuaciones lineales de la siguiente forma: Si el conjunto de entrenamiento de n elementos, $P_1,...,P_n$ (donde $P_i = (P_{i1},...,P_{im})$, con m rasgos), son los vectores de entrada a la red para las clases $C_1,...,C_t$ (no necesariamente todas distintas) podemos determinar los vectores de activación de la capa oculta $I_1,...,I_n$. Por ejemplo para el patrón i:

$$I_i = \left(f\left(\sum_k w_{1k} p_{ik} + b_1\right), f\left(\sum_k w_{2k} p_{ik} + b_2\right), \mathrm{K}, f\left(\sum_k w_{Hk} p_{ik} + b_H\right), 1.0 \right) \quad (5)$$

donde las sumatorias son tomadas para $k=1,...,m$, H es el número de nodos ocultos y w_{ij} es el peso entre la entrada j y el nodo oculto i con un umbral b_i (se adiciona 1.0 para el umbral que este debe ser tratado como un peso más). Para estos vectores es posible formar los vectores presigmoidales de los nodos de salida evaluando la inversa de la función sigmoidal en los valores deseados para cada uno:

$$O_i = \left(f^{-1}(d_{i1}), f^{-1}(d_{i2}), \mathrm{K}, f^{-1}(d_{it}) \right) \quad (6)$$

donde d_{ij} es la salida deseada para el patrón i en la neurona de salida j y f^{-1} es la inversa de la función sigmoidal. Por tanto, la matriz de pesos (y umbrales) W de la capa oculta a la de salida queda contenida en la ecuación:

$$\begin{pmatrix} I_1 \\ \dots \\ I_n \end{pmatrix} \times W = \begin{pmatrix} O_1 \\ \dots \\ O_n \end{pmatrix} \quad (7)$$

donde I es una matriz de orden (número de clusters) x (número de nodos ocultos + 1), W es una matriz (número de nodos ocultos + 1) x (número de nodos de salida) y O es una matriz de orden (número de clusters) x (número de nodos de salida).

El problema con (7) es que en general no existe una solución exacta ya que (número de clusters) ≠ (número de nodos ocultos + 1); por tanto la matriz I no es invertible. Generalmente se tienen más clusters que nodos ocultos. Smyth (1992) propone una Descomposición de Valores Singulares (SVD) para resolver este problema, lo cual resulta en una solución de error cuadrado mínimo. Después de esto, el MLP es entrenado con el algoritmo clásico BP (con todo el conjunto de entrenamiento). Sin embargo, son necesarias muchas menos iteraciones que comenzando con un conjunto inicial de pesos aleatorio. El algoritmo formalizado sería:

Algoritmo para el Diseño de MLP Utilizando Prototipos (DUP).

1. Realizar una clusterización con todos los ejemplos de entrenamiento para determinar el conjunto P de centros de clusters o prototipos.
2. Optimizar la cantidad de hiperplanos según la propuesta de Smyth (1992).
3. Crear la capa oculta del MLP con los hiperplanos que resultaron del paso anterior, así como los pesos de la capa de entrada a la capa oculta según (3) y los umbrales para cada nodo según (4).

4. Determinar los pesos de la capa oculta a la de salida según (7).

3. Consideraciones para la generación de los prototipos

Debido a que la capa oculta no debe ser muy grande, el conjunto de prototipos será mucho más pequeño que el conjunto de entrenamientos. Por tanto, el método usado para sintetizar la información contenida en el conjunto de entrenamiento en forma de unos pocos vectores de referencia adquiere especial importancia, por lo que haremos algunas observaciones.

Para problemas de clasificación, algunos algoritmos supervisados como RCE [11] o LVQ [12] están disponibles. Para problemas de aproximación de funciones continuas también pueden emplearse otros métodos como son: el algoritmo de clusterización k-means, vecino más cercano (NN) o el NeoART, en los cuales el número de prototipos no es fijo, sino determinado por la complejidad de los datos; además estos métodos toman en consideración la probabilidad de distribución de los patrones en el espacio de rasgos y consecuentemente es mucho más satisfactorio que la selección aleatoria del conjunto de entrenamiento.

Una heurística valiosa en la selección del algoritmo para la clusterización es preferir aquellos que generan los prototipos lo más cercano posible a los mínimos y máximos de la función que va a ser aprendida por la red, en el dominio que ésta es aproximada (lo cual ha sido confirmado por muchos otros autores). Uno de estos algoritmos con el que se obtienen muy buenos resultados prácticos está basado en un mecanismo competitivo. Primero, los vectores de referencia son tomados aleatoriamente del conjunto de entrenamiento; además se le proporciona los puntos de la función F que va a ser aprendida. Ya en la fase de aprendizaje, cada patrón de entrada x es presentado al sistema activando el vector de referencia más cercano. Después de un número razonable de ciclos, los prototipos van convergiendo a los puntos máximos y mínimos de F en el conjunto de entrenamiento.

4. Resultados

Las potencialidades del algoritmo han sido exploradas con el conjunto de datos denominados IRIS DATA (ver apéndice), comparándose con el método de inicialización aleatoria. El número de nodos ocultos fue variado para obtener un rango de posibilidades mayor. La tabla 1 muestra algunos de los aspectos de eficiencia que se tomaron en cuenta durante el entrenamiento de los MLP.

Como puede verse, las redes diseñadas utilizando DUP alcanzan sus mejores resultados en igual o menor tiempo que las aleatorias. El tiempo tomado para la clusterización es insignificante comparado con el tiempo de entrenamiento de aplicaciones de MLP como el HP/Apollo DSP 10040, donde son necesarios 1.000 ciclos de BP para ocho nodos ocultos, que es aproximadamente tres horas, mientras que utilizando una técnica de clusterización tomó alrededor de un minuto. El algoritmo DUP ubica al MLP en cercano a un mínimo profundo (como sugieren muchos autores), lo que hace que el proceso de aprendizaje no dependa de las condiciones iniciales ni de parámetros de aprendizaje. Sin embargo el proceso requiere de grandes conjuntos de entrenamiento.

En la figura 1 se representa una red neuronal de una capa oculta y ocho neuronas en dicha capa, obtenida mediante el método DUP. Como puede verse, el número de neuronas de la capa de entrada se corresponde con el número de rasgos de los ejemplos tomados como conjunto de entrenamiento, también el número de neuronas en la capa de salida se corresponde con el total de clases involucradas en el problema. Para determinar el número de unidades de la capa oculta se obtuvieron inicialmente 3 clusters por clase, de los cuales resultaron 27 hiperplanos de separación. Luego de aplicarse el proceso de optimización resultaron ser 8 los hiperplanos necesarios, de ahí el número de nodos ocultos. Los resultados de esta red se pueden ver en la tabla 1.

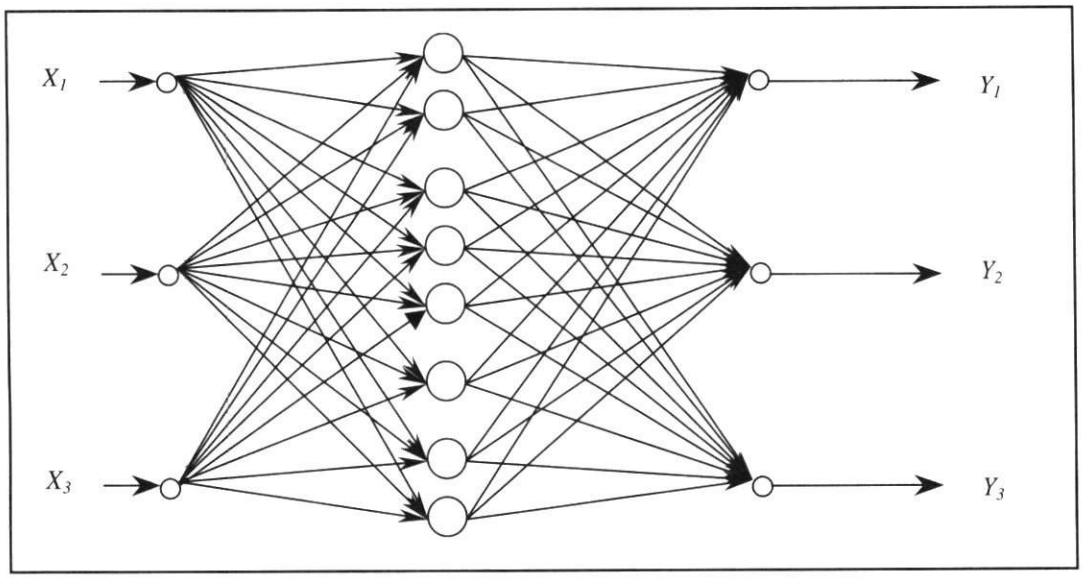

Figura 1 Red neuronal tipo MLP de una capa oculta obtenida mediante el método DUP

Tabla 1 Resultados de los experimentos

Tipo	*Número de nodos ocultos*	*Exactitud del conjunto de entrenamiento antes del entrenamiento*	*Número de ciclos de aprendizaje*	*Error medio al final*	*Mejor error medio al final*	*Exactitud con el conjunto aprendizaje*	*Exactitud con el conjunto de prueba*
DUP	8	68,9%	1.000	-	0,17	89,2%	88,0%
DUP	8	68,9%	2.000	-	0,16	89,5%	87,9%
DUP	14	76,4%	1.000	-	0,14	91,5%	89,1%
DUP	14	76,4%	2.000	-	0,13	92,4%	89,3%
DUP	20	74,4%	1.000	-	0,13	92,2%	89,7%
DUP	20	74,4%	2.000	-	0,11	93,1%	90,3%
Aleatorio	5	8,3%	1.000	0,31	0,30	79,0%	78,5%
Aleatorio	5	8,3%	2.000	0,26	0,25	82,9%	81,1%
Aleatorio	8	8,3%	1.000	0,24	0,24	85,6%	85,2%
Aleatorio	8	8,3%	2.000	0,20	0,20	88,8%	87,5%
Aleatorio	14	8,6%	1.000	0,22	0,21	88,3%	87,2%
Aleatorio	14	8,6%	2.000	0,19	0,17	90,8%	89,6%
Aleatorio	20	8,5%	1.000	0,18	0,16	90,6%	89,0%
Aleatorio	20	8,5%	2.000	0,13	0,13	92,4%	90,4%
Aleatorio	30	8,3%	1.000	0,25	0,21	88,8%	87,6%
Aleatorio	30	8,3%	2.000	0,19	0,17	90,9%	89,6%

5. Conclusiones

Hemos presentado un algoritmo general para el diseño de redes neuronales de tipo MLP que utilizan Backpropagation como algoritmo de aprendizaje. Partiendo de la información que aportan representantes de clases o prototipos obtenidos mediante algún algoritmo de clusterización.

Permite disminuir el tiempo de entrenamiento y hacer una selección óptima del número de nodos ocultos en redes con una capa oculta. Sin embargo el conjunto de entrenamiento debe ser lo suficientemente grande. La cuestión ahora es saber cuántos prototipos deben considerarse para una tarea determinada.

Referencias

1. Sarkar, D. "Methods to speed up error back-propagation learning algorithm". In: *ACM Computing Surveys*. Vol. 27. No. 4. 1995. pp. 519-542.
2. Arai, M. "Bounds on the number of hidden units in binary-valued three-layer neural networks". In: *Neural Networks*. Vol. 6. No. 6. 1993. pp. 855-860.
3. Hirose, Y. *et al.* "Backpropagation algorithm which varies the number of hidden units". In: *Neural Networks*. Vol. 4. No. 1. 1991. pp. 61-66.
4. Huang, S. C. "Bounds on the number of hidden neurons in multilayers perceptrons". In: *IEEE Transactions on Neural Networks*. Vol. 2. No. 1. 1991. pp. 47-55.
5. Irie, B. and Miyake, S. "Capabilities of three-layared perceptrons". In: *IEEE Int. Conf. Neural Networks*. 1988. pp. I-641-I-648.
6. Kung, S. Y. and Hwang, J. N. "An algebraic projection analisys for optimal hidden units size and learning rates in backpropagation learning". In: *IEEE Int. Conf. Neural Networks*. 1988. pp. I-363-I-370.
7. Villiers, J. and Bernard, E. "Backpropagation neural nets with one and two hidden layers". In: *IEEE Transactions on Neural Networks*. Vol. 4. No. 1. 1992. pp. 136-141.
8. Weymaere, N. and Martens, J. P. "A fast and robust learning algorithm for feed forward neural networks". In: *Neural Networks*. Vol. 4. No. 3. 1991. pp. 361-369.
9. Smyth, S. G. "Designing multilayer perceptrons from nearest-neighbor systems". In: *IEEE Transactions on Neural Networks*. Vol. 3. No. 2. 1992. pp. 329-333.
10. Denoeux, T. and Lengellé, R. "Initializing back propagation networks with prototypes". In: *Neural Networks*. Vol. 6. No. 3. 1993. pp. 351-363.
11. Reilly, D. L. *et al.* "A neural model for category learning". In: *Biological Cybernetics*. 1982. pp. 45, 35-41.
12. Kohonen, T. *Self-organization and associative memory*. Berlín, Springer-Verlag, 1987.

Apéndice

Conjunto de entrenamiento Iris Data. Posee tres clases con 50 ejemplos cada una y cuatro rasgos por ejemplo.

Clase No. 1				*Clase No. 2*				*Clase No. 3*			
5,0	3,5	1,4	0,2	7,0	3,2	4,7	1,4	6,3	3,3	6,0	2,5
4,9	3,0	1,4	0,2	6,4	3,2	4,5	1,5	5,8	2,7	5,1	1,9
4,7	3,2	1,3	0,2	6,9	3,1	4,9	1,5	7,1	3,0	5,9	2,1
4,6	3,1	1,5	0,2	5,56	2,3	4,0	1,3	6,3	2,9	5,6	1,8
5,0	3,6	1,4	0,2	6,5	2,8	4,6	1,5	6,5	3,0	5,8	2,2
5,4	3,9	1,7	0,4	5,7	2,8	4,5	1,3	7,6	3,0	6,6	2,1
4,6	3,4	1,4	0,3	6,3	3,3	4,7	1,6	4,9	2,5	4,5	1,7
5,0	3,4	1,5	0,2	4,9	2,4	3,3	1,0	7,3	2,9	6,3	1,8
4,4	2,9	1,4	0,2	6,6	2,9	4,6	1,3	6,7	2,5	5,8	1,8
4,9	3,1	1,5	0,1	5,2	2,7	3,9	1,4	7,2	3,6	6,1	2,5
5,4	3,7	1,5	0,2	5,0	2,0	3,5	1,0	6,5	3,2	5,1	2,0
4,8	3,4	1,6	0,2	5,9	3,0	4,2	1,5	6,4	2,7	5,3	1,9
4,8	3,0	1,4	0,1	6,0	2,2	4,0	1,0	6,8	3,0	5,5	2,1
4,3	3,0	1,1	0,1	6,1	2,9	4,7	1,4	5,7	2,5	5,0	2,0
5,8	4,0	1,2	0,2	5,6	2,9	3,6	1,3	5,8	2,8	5,1	2,4
5,7	4,4	1,5	0,4	6,7	3,1	4,4	1,4	6,4	3,2	5,3	2,3
5,4	3,9	1,3	0,4	5,6	3,0	4,5	1,5	6,5	3,0	5,5	1,8

Clase No. 1				*Clase No. 2*				*Clase No. 3*			
5,1	3,5	1,4	0,3	5,8	2,7	4,1	1,0	7,7	3,8	6,7	2,2
5,7	3,8	1,7	0,3	6,2	2,2	4,5	1,5	7,7	2,6	6,9	2,3
5,1	3,8	1,5	0,3	5,6	2,5	3,9	1,1	6,0	2,2	5,0	1,5
5,4	3,4	1,7	0,2	5,9	3,2	4,8	1,8	6,9	3,2	5,7	2,3
5,1	3,7	1,5	0,4	6,1	2,8	4,0	1,3	5,6	2,8	4,9	2,0
4,6	3,6	1,0	0,2	6,3	2,5	4,9	1,5	7,7	2,8	6,7	2,0
5,1	3,3	1,7	0,5	6,1	2,8	4,7	1,2	6,3	2,7	4,9	1,8
4,8	3,4	1,9	0,2	6,4	2,9	4,3	1,3	6,7	3,3	5,7	2,1
5,0	3,0	1,6	0,2	6,6	3,0	4,4	1,4	7,2	3,2	6,0	1,8
5,0	3,4	1,6	0,4	6,8	2,8	4,8	1,4	6,2	2,8	4,8	1,8
5,2	3,5	1,5	0,2	6,7	3,0	5,0	1,7	6,1	3,0	4,9	1,8
5,2	3,4	1,4	0,2	6,0	2,9	4,5	1,5	6,4	2,8	5,6	2,1
4,7	3,2	1,6	0,2	5,7	2,6	3,5	1,0	7,2	3,0	5,8	1,6
4,8	3,1	1,6	0,2	5,5	2,4	3,8	1,1	7,4	2,8	6,1	1,9
5,4	3,4	1,5	0,4	5,5	2,4	3,7	1,0	7,9	3,8	6,4	2,0
5,2	4,1	1,5	0,1	5,8	2,7	3,9	1,2	6,4	2,8	5,6	2,2
5,5	4,2	1,4	0,2	6,0	2,7	5,1	1,6	6,3	2,8	5,1	1,5
4,9	3,1	1,5	0,1	5,4	3,0	4,5	1,5	6,1	2,6	5,6	1,4
5,0	3,2	1,2	0,2	6,0	3,4	4,5	1,6	7,7	3,0	6,1	2,3
5,5	3,5	1,3	0,2	6,7	3,1	4,7	1,5	6,3	3,4	5,6	2,4
4,9	3,1	1,5	0,1	6,3	2,3	4,4	1,3	6,4	3,1	5,5	1,8
4,4	3,0	1,3	0,2	5,6	3,0	4,1	1,3	6,0	3,0	4,8	1,8
5,1	3,4	1,5	0,2	5,5	2,5	4,0	1,3	6,9	3,1	5,4	2,1
5,0	3,5	1,3	0,3	5,5	2,6	4,4	1,2	6,7	3,1	5,6	2,4
4,5	2,3	1,3	0,3	6,1	3,0	4,6	1,4	6,9	3,1	5,1	2,3
4,4	3,2	1,3	0,2	5,8	2,6	4,0	1,2	5,8	2,7	5,1	1,9
5,0	3,5	1,6	0,6	5,0	2,3	3,3	1,0	6,8	3,2	5,9	2,3
5,1	3,8	1,9	0,4	5,6	2,7	4,2	1,3	6,7	3,3	5,7	2,5
4,8	3,0	1,4	0,3	5,7	3,0	4,2	1,2	6,7	3,0	5,2	2,3
5,1	3,8	1,6	0,2	5,7	2,9	4,2	1,3	6,3	2,5	5,0	1,9
4,6	3,2	1,4	0,2	6,2	2,9	4,3	1,3	6,5	3,0	5,2	2,0
5,3	3,7	1,5	0,2	5,1	2,5	3,0	1,1	6,2	3,4	5,4	2,3
5,0	3,3	1,4	0,2	5,7	2,8	4,1	1,3	5,9	3,0	5,1	1,8