Revista Cubana de Ciencias Informáticas
Vol. 10, No. 4, Octubre-Diciembre, 2016
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 154-170

Tipo de artículo: Artículo original
Temática: Inteligencia artificial
Recibido: 05/06/20116 | Aceptado: 28/08/2016

Una evaluación del algoritmo LVQ en una colección de texto

An evaluation of LVQ algorithm in a textual collection

Alain Guerrero Enamorado[1*], Daimerys Ceballos Gastell[1]

[1] Universidad de las Ciencias Informáticas, Carretera San Antonio, Km 2 ½ Torrens, Boyeros. La Habana. Cuba. dceballo@uci.cu

*Autor para correspondencia: alaing@uci.cu

Resumen

Este trabajo se realizó con el objetivo de obtener información empírica sobre el comportamiento del algoritmo LVQ (Aprendizaje por Cuantificación Vectorial) en la tarea de clasificación de información sobre 24 versiones de una colección de noticias creada previamente, en las cuales se varió la cantidad de términos utilizando el método, Ganancia de Información, para la selección de atributos. Se logra determinar el comportamiento de los algoritmos frente a los efectos de reducción de dimensionalidad. Se encuentra un punto, donde el compromiso entre pérdida de información e incorporación de información irrelevante, logra buenos resultados como promedio entre los distintos algoritmos evaluados.

Palabras Clave: aprendizaje por cuantificación vectorial, clasificación, redes neuronales

Abstract

The aim of this work was to obtain empirical information on the behavior of the LVQ (Learning Vector Quantization) algorithm in the task of information classification. We developed an algorithm evaluation on 24 versions of a news-collection previously created. In this collection, we varied the number of terms using the method, Information Gain, to select the attributes. The behavior of algorithms against the effects of dimensionality reduction is determined. We found a point, where the compromise between loss of information and incorporation of irrelevant information, has achieved good results.

Keywords: *learning vector quantification, classification, neural networks*

Introducción

Cuando de clasificar información se trata un método simple para hacerlo pudiera ser encontrar una función lineal que devuelva un valor que represente la membresía a una clase determinada, utilizando para ello como variables de entrada los atributos predictores del ejemplo que se quiere clasificar. Sin embargo, en los problemas del mundo real no existen la mayoría de las veces funciones lineales capaces de hacer esto. Cuando esto ocurre, nos encontramos frente a problemas no-lineales donde una posible solución puede ser utilizar técnicas de inteligencia artificial para estimar este tipo de funciones.

En este trabajo se realiza un estudio experimental del algoritmo de clasificación Aprendizaje por Cuantificación Vectorial (LVQ) propuesto por Tuevo Kohonen en el trabajo (Kohonen 1988).

Con este trabajo se persigue como objetivo fundamental realizar una evaluación del LVQ sobre 24 versiones distintas de una colección textual construida por los autores con noticias en inglés tomadas de medios de prensa y clasificadas en las categorías: cultura, economía, ciencia-y-tecnología y deportes. Las 24 versiones se crearon a partir de aplicar el método de ganancia de información para reducir la dimensionalidad de la colección original de 21731 términos.

En la sección 0 se detalla el algoritmo en sus principales versiones, posteriormente se exponen las principales características del resto de los algoritmos utilizados para la experimentación. El algoritmo LVQ también puede entrenarse sin clases de salida utilizando un aprendizaje no-supervisado para aplicarlo en la tarea de agrupamiento. Desde hace algunos años atrás, como se vislumbra en los artículos (Bezdek & Pal 1995; Karayiannis & Pai 1996; Karayiannis 1997; Karayiannis 1999; Pal et al. 1993) pueden encontrarse este tipo de aplicaciones que no se tratan aquí. Sin embargo, el ámbito del presente trabajo es básicamente la aplicación del LVQ solamente en la tarea de clasificación.

El algoritmo LVQ ha sido y sigue siendo sometido a una gran cantidad de investigaciones por parte de la comunidad científica internacional. Algunas de las extensiones que se le han realizado, toman como punto de partida la sustitución de la medida de distancia por métricas más generales tales como: medida euclidiana pesada (Hammer & Villmann 2002), matriz de relevancia adaptativa (Schneider et al. 2009), métrica pseudo-euclidiana (Hammer et al. 2011) y métricas en el espacio del núcleo característico (Qin & Suganthan 2004). Muchas también han sido las aplicaciones del LVQ en varios campos como por ejemplo: en el procesamiento de señales e imágenes (Bashyal & Venayagamoorthy 2008; Blume & Ballard 2007), en la industria (Ahn & Nguyen 2007; Bassiuny et al. 2007), en la medicina (Anagnostopoulos et al. 2001; Chen 2012; Dieterle et al. 2003; Dutta et al. 2001) y muchos otros. Una lista más exhaustiva puede encontrarse en línea en la base de datos (Neural Networks Research Centre 2007). Como se puede constatar muchas han sido las investigaciones realizadas por la comunidad científica sobre el algoritmo LVQ, motivo por el cual los autores se motivaron a realizar una evaluación del mismo para una colección de noticias como posible

Revista Cubana de Ciencias Informáticas
Vol. 10, No. 4, Octubre-Diciembre, 2016
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 154-170
aplicación en la clasificación de texto. En la sección 0 se mencionan las herramientas y métodos. Finalmente en la sección 0 se exponen los resultados obtenidos y se dan algunas recomendaciones para la continuidad de este trabajo.

Materiales y métodos

En esta sección se detallan los algoritmos empleados en los diferentes experimentos realizados, partiendo de la explicación detallada del algoritmo LVQ en sus variantes fundamentales y exponiendo las principales características del resto de los algoritmos utilizados para la experimentación.

Aprendizaje por cuantificación vectorial (LVQ)

Tuevo Kohonen presentó un modelo de red neuronal (Kohonen 1988) con capacidad para formar mapas de características de manera similar a como ocurre en el cerebro. Este modelo parte de dos variantes: el aprendizaje por cuantificación vectorial (LVQ, Learning Vector Quantization) y los mapas auto-organizados (SOM, Self Organizing Map). En ambos casos se construyen mapas topológicos donde son agrupadas de cierta manera las entidades con características similares. Ambos algoritmos utilizan un aprendizaje competitivo reforzado, distinguiéndose una etapa de entrenamiento y otra de explotación (Kohonen 1990; Kohonen 2001). Una de las principales diferencias que existen entre el algoritmo LVQ y su precursor SOM radica en que este último agrupa las instancias de acuerdo a su similitud, mientras que en el primero una vez que fueron agrupadas se les asigna además una clase. Esta última característica del LVQ es la que permite que sea utilizado en la tarea de clasificación.

El modelo LVQ utiliza una red de dos capas como el mostrado en la Figura 1, sus conexiones se realizan hacia delante. Los pesos de conexión se representan mediante una matriz de pesos W según cada capa de la red neuronal.

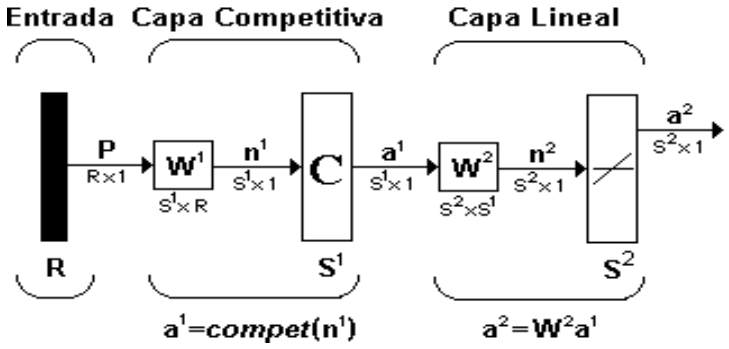

Figura 1: Diagrama en bloques de una red LVQ

El proceso de entrenamiento comienza con la inicialización de los pesos sinápticos. Dicha inicialización se puede realizar de varias formas, entre ellas están: pesos nulos, pesos aleatorios de pequeño valor absoluto, o pesos con un valor de partida predefinido. Luego se escoge una instancia del conjunto de entrenamiento y se presenta a la entrada de la

red. Cada vector prototipo (vector fila de la matriz de pesos W^1, o vector formado con el conjuntos de todos los pesos que unen cada atributo de entrada con una neurona en particular de esta capa competitiva), de la capa competitiva calcula su distancia con respecto al vector de entrada en base a alguna función de similitud, como por ejemplo: producto interno, función coseno, distancia *hamming*, distancia euclidiana, etc. El criterio de medida utilizado en este trabajo fue la distancia euclidiana definida por la ecuación (1).

$$dist_{eucl}(P, W_i^1) = \sqrt{\sum_{j=1}^{R}(r_j - w_{ij}^1)^2} \qquad (1)$$

Después se establece una competición entre los vectores prototipos y el vector de entrada para determinar cuál es el más cercano al vector de entrenamiento el cual será proclamado ganador de dicha competición, es decir, se determina la neurona ganadora, cuya distancia sea la menor de todas. Así la salida de la primera capa de la red LVQ se define por la ecuación (2).

$$a^1 = Compet(n^1) \qquad (2)$$

Donde a^1 es la salida y n^1 es la entrada neta a la función de transferencia, definida por la ecuación (3).

$$n^1 = \begin{bmatrix} dist_{eucl}(P, W_1^1) \\ dist_{eucl}(P, W_2^1) \\ \vdots \\ dist_{eucl}(P, W_{S^1}^1) \end{bmatrix} \qquad (3)$$

La función $Compet(.)$ es una función de transferencia que encuentra el índice i-ésimo de la neurona con la entrada neta con mayor similitud con la entrada actual y fija su salida en uno, el resto de las neuronas devuelven como salida cero (McClelland et al. 1987). Dicho de otra forma, la neurona cuyo vector de pesos esté más cercano al vector de entrada tendrá salida 1 y el resto de las neuronas tendrán salida 0; la salida no cero representa una sub-clase, muchas neuronas (subclases), conforman una clase. La segunda capa de la red LVQ es usada para combinar varias subclases dentro de una sola clase, esto es realizado por la matriz de pesos W^2. Las columnas de W^2 representan las subclases y las filas representan las clases, W^2 tiene un solo 1 en cada columna, todos los demás elementos son cero, la fila de la columna en la cual se presenta el 1 índica cuál es la clase a la que la subclase pertenece. Dicho de otra forma si $W_{ki}^2 = 1$; entonces la subclase i pertenece a la clase k. Una propiedad importante de esta red, es que el proceso de combinar subclases para formar clases, permite la creación de clases más complejas. Una sola capa competitiva estándar tiene la limitación de que puede crear solo regiones de decisión convexas; la red LVQ soluciona esta limitación (Kohonen 2001).

Para generar la matriz W^2, antes de que se inicie el aprendizaje, cada neurona en la segunda capa es asignada a una neurona de salida; por lo general igual número de neuronas ocultas son conectadas a cada neurona de salida, para que

Revista Cubana de Ciencias Informáticas
Vol. 10, No. 4, Octubre-Diciembre, 2016
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 154-170

cada clase pueda ser conformada por el mismo número de regiones convexas. Una vez que ha sido definida nunca será alterada durante el proceso de entrenamiento que solo modificará los pesos de W^1. Para obtener la salida final se calcula $a^2 = W^2 * a^1$ que indicará a quien está siendo asignado el vector de entrada P presentado inicialmente a la red. Se procede entonces a la actualización de los pesos sinápticos de la neurona ganadora por medio de la regla de Kohonen (Kohonen 1990) que es empleada para mejorar solo la capa oculta de la red LVQ.

Puesto que los vectores prototipos y los patrones de entradas están etiquetados con las clases a las que pertenecen, las correcciones son del tipo premio o castigo. Si se acertó en la clasificación (la clase del vector prototipo ganador y la clase del vector de entrada coinciden) se premia al vector prototipo acercándolo aún más al vector de entrada. Por el contrario, si la clasificación no se ha realizado correctamente (la clase seleccionada por el vector prototipo ganador es diferente a la clase del vector de entrada), se castiga al vector prototipo alejándolo del vector de entrada. La actualización de los pesos para el instante $t + 1$ se realiza según la ecuación (4).

$$W_i^1(t+1) = W_i^1(t) + \alpha(t) * dist_{eucl}\left(P(t), W_i^1(t)\right) \quad (4)$$

Donde $\alpha(t)$ es la tasa de aprendizaje, la misma se utiliza de tres formas posibles en la ecuación (4):

1. $\alpha(t)$ con signo positivo si $W_i^1(t)$ y $P(t)$ pertenecen a la misma clase, en este caso se premia la neurona ganadora.
2. $\alpha(t)$ con signo negativo si $W_i^1(t)$ y $P(t)$ pertenecen a clases diferentes, en este caso se castiga la neurona ganadora.
3. $\alpha(t) = 0$, es decir $W_i^1(t+1) = W_i^1(t)$, para el caso de cualquier otro vector prototipo $W_i^1(t)$ distinto del ganador.

Este parámetro $\alpha(t)$ o tasa de aprendizaje es utilizado por muchas redes neuronales en su fase de entrenamiento para controlar la velocidad de convergencia de los pesos sinápticos. Resulta de gran importancia la elección de un valor adecuado para la tasa de aprendizaje que permita hallar un punto de equilibrio entre la velocidad de convergencia y la estabilidad final de los vectores prototipo. Una $\alpha(t)$ cercana a *0*, hace que el aprendizaje sea lento pero asegura que cuando un vector prototipo haya alcanzado una clase se mantenga allí indefinidamente; sin embargo valores altos de $\alpha(t)$ (cercanos a *1*) hacen que el acercamiento del vector prototipo al vector de entrada sea muy rápido, pero tiene como desventaja que los pesos tendrán grandes oscilaciones provocando muchas veces inestabilidad en el proceso de convergencia. Otra posibilidad es utilizar un $\alpha(t)$ adaptativo, se inicializa con algún valor no muy alto, por ejemplo *0.3*; y se decrementa a medida que avanza el entrenamiento según alguna función, de manera que al final del proceso su valor

Revista Cubana de Ciencias Informáticas
Vol. 10, No. 4, Octubre-Diciembre, 2016
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 154-170

sea muy cercano a *0*. Normalmente, por motivos computacionales, se suele elegir una función lineal, monótona decreciente y que tome valores en el intervalo (*0; 1*).

Kohonen y sus colaboradores han desarrollado diferentes versiones del algoritmo inicial LVQ, en este trabajo se utilizan las versiones LVQ1, LVQ2.1 y LVQ3 implementaciones disponibles en la herramienta para minería de datos KEEL[1] (Alcalá-Fdez et al. 2008) (*Knowledge Extraction based on Evolutionary Learning*).

Variante LVQ 2.1

El cambio fundamental respecto al algoritmo original radica en la forma de actualizar los pesos. En esta versión en lugar de seleccionar un único vector prototipo ganador son seleccionados los dos (W_i^1 y W_j^1) más próximos al vector de entrada (X), pero con la condición que uno de ellos pertenezca a la clase del vector de entrada y el otro no. Además, el vector de entrada debe quedar situado cercano al punto medio de la ventana que se forma entre estos dos vectores prototipo. Definiendo:

- s como el ancho de la ventana (valores recomendados entre 0.2 y 0.3) y
- d_i y d_j distancias euclidianas desde el vector de entrada hasta cada uno de los vectores prototipo ganadores.

Se dice que este vector se encuentra dentro de la ventana si satisface la ecuación (5).

$$mínimo\left(\frac{d_j}{d_i}, \frac{d_i}{d_j}\right) > \frac{1-s}{1+s} \quad (5)$$

En esta versión las ecuaciones de actualización de los pesos quedan como en 6 y 7.

$$W_i^1(t+1) = W_i^1(t) + \alpha(t) * dist_{eucl}\left(X(t), W_i^1(t)\right) \quad (6)$$

$$W_j^1(t+1) = W_j^1(t) - \alpha(t) * dist_{eucl}\left(X(t), W_j^1(t)\right) \quad (7)$$

Con esta variante del algoritmo se logra un efecto doble puesto que en cada iteración se premia al mejor vector prototipo que pertenece a la misma clase que el vector de entrada al mismo tiempo que se castiga al vector prototipo más cercano que pertenece a una clase distinta a la del vector de entrada.

Variante LVQ 3

En esta nueva versión se realiza una combinación entre las dos variantes anteriores LVQ 1 y LVQ 2.1. Al igual que en la versión anterior son seleccionados los dos vectores prototipo (W_i^1 y W_j^1) más próximos al vector de entrada (X). Ahora si W_i^1 pertenece a la clase del vector de entrada y W_j^1 no pertenece a la misma clase que X, y además ambos

[1] Disponible en http://www.keel.es/

Revista Cubana de Ciencias Informáticas
Vol. 10, No. 4, Octubre-Diciembre, 2016
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 154-170

quedan definidos en la ventana como en el LVQ 2.1 las modificaciones a los pesos sinápticos se realizan exactamente igual a como se definieron por las ecuaciones 6 y 7. Pero si por el contrario ambos vectores prototipo pertenecen a la misma clase que X entonces ambos son premiados según las siguientes dos nuevas ecuaciones 8 y 9.

$$W_i^1(t+1) = W_i^1(t) + \varepsilon * \alpha(t) * dist_{eucl}\left(X(t), W_i^1(t)\right) \quad (8)$$

$$W_j^1(t+1) = W_j^1(t) - \varepsilon * \alpha(t) * dist_{eucl}\left(X(t), W_j^1(t)\right) \quad (9)$$

Donde ε toma valores en el intervalo [0.1; 0.5].

Trabajos relacionados

En este trabajo se hace una comparación del algoritmo LVQ frente a cuatro técnicas de las más reconocidas por la comunidad científica (Wu et al. 2007). En particular se realiza la comparación contra los algoritmos: árboles de decisión C4.5 (Quinlan 1993), Bayesiano Ingenuo o NB (Domingos & Pazzani 1997; John & Langley 1995; Maron 1961), K-vecinos más cercanos o KNN (Fix & Hodges 1951; Cover et al. 1967; Wang et al. 2007) y las Máquinas de Soporte Vectorial o SVM (Cortes et al. 1995). En todos los casos se utilizaron las implementaciones de los algoritmos presentes en la herramienta KEEL (Alcalá-Fdez et al. 2008).

LVQ1 optimizado (OLVQ1)

Entre muchas variantes del LVQ, está el caso del OLVQ1, este utiliza la misma ecuación 4 para actualizar los pesos W_i^1 pero en lugar de utilizar un $\alpha(t)$ constante se utiliza un $\alpha_i(t)$ variable en el tiempo. De esta manera si $W_i^1(t)$ y $P(t)$ pertenecen a la misma clase se utiliza la ecuación 10 para actualizar $\alpha_i(t)$. En cambio si $W_i^1(t)$ y $P(t)$ pertenecen a clases diferentes se utiliza la ecuación 11 para actualizar $\alpha_i(t)$. En caso de que se trate de un vector prototipo $W_i^1(t)$ distinto del ganador se mantiene constante $\alpha_i(t+1) = \alpha_i(t)$.

$$\alpha_i(t+1) = \frac{\alpha_i(t)}{1+\alpha_i(t)} \quad (10)$$

$$\alpha_i(t+1) = \frac{\alpha_i(t)}{1-\alpha_i(t)} \quad (11)$$

Sin embargo, hasta el momento la herramienta KEEL no dispone de la implementación del OLVQ1, por lo que se deja para futuros trabajos la posibilidad de extender el KEEL con su implementación para así incluir y enriquecer aún más la evaluación experimental del algoritmo LVQ de este artículo.

En la bibliografía no existen muchas referencias sobre cual algoritmo escoger para un problema en cuestión, el mismo Kohonen sostiene que las cuatro variantes dan resultados muy similares (Kohonen 2001). Sin embargo en el ámbito de este trabajo se obtienen algunas diferencias en el comportamiento de las distintas versiones las cuales se discuten en las

próximas secciones 0 y 0.

Árboles de decisión (C4.5)

Este algoritmo genera reglas de clasificación en forma de árboles de decisión partiendo de un conjunto de entrenamiento, dicho árbol se construye de arriba hacia abajo y en cada paso de ejecución del algoritmo se realiza una prueba en cada nodo (partiendo del nodo raíz) para determinar cuál separa mejor los ejemplos de la colección de acuerdo a la clase a la que pertenecen.

Este algoritmo surgió inicialmente como una mejora al ID3 (Quinlan 1986) para permitir información incompleta o valores omitidos en los datos, además maneja atributos continuos, utiliza una métrica para la selección de los atributos que maximiza la ganancia de información y finalmente poda el resultado partiendo del principio de mínima longitud de descripción o principio MDL (Rissanen 1978).

Se puede configurar en KEEL con los siguientes parámetros:

- *Prune*: este parámetro permite activar o desactivar el mecanismo de poda de los árboles.
- *Confidence*: define el nivel mínimo de confianza que debe tener una hoja para mantenerse dentro del árbol.
- *minItemsets*: define el número mínimo de instancias por hojas.

Bayesiano Ingenuo (NB)

Este algoritmo se basa en la aplicación del teorema de probabilidad condicional de Bayes (Bayes 1763), al mismo tiempo que se asume independencia entre los atributos predictores para poderse aplicar dicho teorema. El algoritmo calcula la probabilidad condicional de cada ejemplo pertenecer a cada clase, entonces se asigna como clase de un nuevo ejemplo la de mayor probabilidad condicional calculada anteriormente. Este tipo de clasificador a pesar de su sencillez ha demostrado comportarse muy bien en muchos problemas incluso aun cuando no se cumple la asunción de independencia de los atributos de entrada.

Este modelo se puede configurar en KEEL sin ningún parámetro, aunque la implementación solo permite atributos nominales, por lo cual se utilizó previamente en las colecciones numéricas una discretización de frecuencia uniforme (Liu et al. 2002) con 10 intervalos.

K-vecinos más cercanos (KNN)

Método de clasificación supervisada que permite estimar una función de clasificación partiendo de los atributos predictores. La información para estimar la función se obtiene del conjunto de los *k* vecinos más cercanos. Cada ejemplo de la colección es un punto en un espacio multidimensional de los atributos descriptores. Utilizando una medida de distancia se estima cuan cerca se encuentra el ejemplo actual del resto, tomando entonces las clases a las que pertenecen

los *k* más cercanos se tiene un voto para asignar la clase del ejemplo en cuestión. Se utiliza generalmente la distancia euclidiana.

Este modelo se puede configurar en KEEL con los siguientes parámetros:

- *k*: número de vecinos a encuestar.
- *Distance function*: se implementan 3 posibilidades en KEEL:
  - *Euclidean*: con atributos normalizados.
  - *Heterogeneous Value Difference Metric* o HVDM (Wilson & Martinez 2000): esta utiliza la distancia euclidiana para atributos numéricos y la VDM (Stanfill & Waltz 1986) para atributos nominales.
  - *Manhattan*: la distancia entre dos puntos es el valor absoluto de la suma de las diferencias de las coordenadas de estos.

Máquinas de Soporte Vectorial (SVM)

Realiza la tarea de clasificación por medio de encontrar un hiperplano que maximiza el margen entre dos clases. Los vectores (ejemplos de la colección) que definen el hiperplano son los vectores de soporte. El algoritmo inicia con definir un hiperplano óptimo que maximice el margen, entonces los datos son mapeados a un espacio de mayor dimensión, por medio de una función de *kernel*, donde es más fácil encontrar superficies de decisión lineales. En el caso ideal este algoritmo logra generar un hiperplano que separa completamente las clases sin solapamiento. No obstante, el caso ideal casi nunca se logra quedando un ligero solapamiento entre las clases que se traduce en errores de clasificación. Por lo anterior el algoritmo lo que busca es encontrar el hiperplano que maximiza el margen y minimiza los errores de clasificación. Para el caso multi-clase SVM utiliza un esquema de votación.

Este modelo se puede configurar en KEEL con los siguientes parámetros:

- *KernellType*: función de *kernel* a utilizar para transformar los datos.
- *C*: parámetro que define el costo del error de clasificación.
- *Degree*: grados de libertad de la función de *kernel*.
- *Gamma*: parámetro gamma de la función de *kernel*.
- *Coef0*: parámetro *coef0* en la función de *kernel*.
- *Shrinking*: reduce el tamaño del problema de optimización dejando de considerar algunas de las variables.

Resultados y discusión

En esta sección se presentan los resultados empíricos del proceso de evaluación de los algoritmos. Fue utilizada la

exactitud predictiva como métrica para la evaluación del desempeño de todos los algoritmos involucrados.

Colección de noticias.

Se construyó una colección de entrenamiento de textos en inglés compuesta por documentos de noticias de prensa, extraídas de los siguientes medios:

- Granma, disponible en: http://www.granma.cu/
- Juventud Rebelde, disponible en: http://www.juventudrebelde.cu/
- Agencia Cubana de noticias (ACN), disponible en: http://www.cubanews.ain.cu/
- Prensa Latina, disponible en: http://www.prensa-latinaenglish.com/
- Cubarte, disponible en: http://www.cubarte-english.cult.cu/
- CNN, disponible en: http://edition.cnn.com/

Para representar los documentos se utilizó el Modelo de Espacio Vectorial (MEV) (Salton et al. 1975), pero antes de construirlo, todos los documentos fueron pre-procesados. Para ello se convirtieron todas las mayúsculas a minúsculas, se eliminaron los números, se aplicó un filtro para reducir la cantidad de letras consecutivas repetidas más de tres veces en una palabra, se eliminaron las palabras vacías que se encuentran en la lista de parada del sistema de recuperación de información SMART[2]. También se eliminaron las 1000 palabras más frecuentes del idioma inglés[3] debido a que estas palabras tienen una gran probabilidad de aparecer en casi todos los documentos y haría el idf (*inverse document frequency*) muy cercano a cero haciendo que la representación de esos términos en el espacio vectorial igual a cero. Luego, a cada término se le aplicó la función de *stemming* de Porter[4] ampliamente utilizada, que permite extraer los sufijos y prefijos comunes de palabras literalmente diferentes pero con una raíz común que pueden ser consideradas como un sólo término. Tras este proceso se obtuvieron un total de *21731* palabras.

En el MEV cada documento se representa mediante un vector de términos. Cada término lleva asociado un coeficiente o peso que trata de expresar la importancia o grado de representatividad de ese término en ese vector o documento. De esta manera, un conjunto de m documentos y de n términos sería una matriz de n filas por m columnas. El valor de cada componente del vector representa la importancia relativa de ese término en el documento. Una aproximación

[2] Disponible en ftp://ftp.cs.cornell.edu/pub/smart

[3] Disponibles en http://members.unine.ch/jacques.savoy/clef/index.html

[4] Disponible en http://www.tartarus.org/~martin/PorterStemmer

Revista Cubana de Ciencias Informáticas
Vol. 10, No. 4, Octubre-Diciembre, 2016
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 154-170

común para el pesado de términos usa la frecuencia de ocurrencia de una palabra determinada en el documento para representar las componentes del vector. Para calcular los pesos de los términos se usa la ecuación estándar $tf * idf$, donde tf (*term frequency*) identifica la frecuencia del término en el documento, y idf (*inverse document frequency*) es el inverso de la frecuencia del documento definida en la ecuación 12.

$$idf_i = \log_2 \frac{M}{df_i} \quad (12)$$

Donde la frecuencia del documento df_i identifica al número de documentos de la colección en el que se encuentra presente el término y M es el número total de documentos en la colección. Finalmente se creó una colección con 800 documentos, 21 731 términos y 4 clases distintas con 200 documentos por cada una.

Para obtener las 24 versiones distintas partiendo de la colección anterior, se aplicó desde la herramienta Weka (Hall et al. 2009) un filtro supervisado de selección de atributos en el que se utilizó el método de ganancia de información para reducir la dimensionalidad de la colección original. Seleccionándose: 20, 40, 60, 80, 100, 200, 300, 400, 500, 600, 700, 800, 900,1000, 1100, 1200, 1300, 1400, 1500, 1600, 1700, 1800, 1900 y 2000 términos.

Herramientas utilizadas en la experimentación

Se ejecutaron todas las simulaciones con la herramienta KEEL (Alcalá-Fdez et al. 2011), los experimentos se ejecutaron en un procesador Dual Core AMD-300 a 1.3GHz, 4GB de memoria RAM y Sistema Operativo Windows 8. En general se utilizaron las configuraciones por defecto de KEEL excepto en el caso del algoritmo KNN en el cual se utilizó k = 5 y en los algoritmos LVQ donde se utilizaron una cantidad de iteraciones de 500 en lugar de 100. Las configuraciones de cada caso se pueden consultar en la Tabla .

Tabla 1. Valores de configuración de los algoritmos utilizados con la herramienta KEEL.

Algoritmo	Configuraciones
LVQ1	*number_of_iterations = 500* *percentage_respect_training_size = 20* *alpha_0 = 0.1*
LVQ2.1	*number_of_iterations = 500* *percentage_respect_training_size = 20* *alpha_0 = 0.1 / windowWidth = 0.2*
LVQ3	*number_of_iterations = 500* *percentage_respect_training_size = 20* *alpha_0 = 0.1 / windowWidth = 0.2* *epsilon = 0.1*
SVM	*KERNELtype = POLY* *C = 100.0 / eps = 0.001 / degree = 1* *gamma = 0.01 / coef0 = 0.0 / nu = 0.1 p = 1.0 / shrinking = 1*
C45	*pruned = TRUE* *confidence = 0.25*

	instancesPerLeaf = 2
KNN	*K Value = 5 / Dist. Funct. = Euclidean*
NB	*Uniform Frequency Discretizer* *numIntervals = 10*

Resultados experimentales.

Para la experimentación se utilizó el procedimiento de validación cruzada con 10 particiones y además 5 ejecuciones con semillas aleatorias distintas cada vez, para un total de 50 experimentos (en cada par algoritmo-colección), los resultados de la métrica exactitud predictiva (cantidad de ejemplos clasificados correctamente sobre el total de ejemplos) fueron promediados para medir la competitividad de cada algoritmo en cada colección de datos. El experimento se diseñó para escoger una mínima cantidad de términos que permitieran reducir la dimensionalidad de la colección de noticias creada inicialmente como se describió en la sección 0.

Tras la ejecución de los 7 algoritmos (LVQ1, LVQ2.1, LVQ3, SVM, C45, KNN y NB) en las 24 versiones de la colección de datos de noticias podemos mostrar los resultados obtenidos en la

Tabla en valor porcentual de la exactitud predictiva. En la misma por razones de espacio solo se colocaron en la primera columna de la izquierda la cantidad de términos que define la versión de la colección de noticias utilizada en ese caso y se añadió una columna adicional al final con el promedio entre todos los algoritmos con el fin de delimitar la cantidad de términos mínimos necesarios para lograr una correcta clasificación en la colección de noticias. La zona donde se obtienen los mejores resultados de clasificación como promedio se encuentra entre los 400 y los 900 términos, y el primer punto donde se logran los mejores resultados es en el valor de 600 términos donde se alcanza un promedio de 89.1%.

Tabla 2 Exactitud predictiva (%) de cada algoritmo en la colección de noticias variando la cantidad de términos desde 20 hasta 2000.

# de Términos	LVQ1	LVQ 2.1	LVQ3	SVM	C45	KNN	NB	Promedio
20	77,5	74,7	78,2	84,5	85,8	75,5	86,9	78,7
40	80,2	77,3	79,2	89,4	91,0	82,5	92,0	81,5
60	78,4	77,5	78,3	88,0	89,1	79,6	91,6	80,6
80	79,0	76,8	77,9	88,8	89,4	79,3	92,3	80,6
100	79,5	76,7	79,1	88,5	89,3	78,4	92,3	80,9
200	85,8	78,9	85,4	91,0	87,4	77,9	92,1	85,3
300	87,1	79,1	84,0	92,6	87,8	69,0	91,9	85,7
400	89,0	82,0	87,7	93,4	86,9	74,6	91,8	88,0
500	89,3	83,1	87,7	94,3	86,8	73,1	91,5	88,6
600	90,0	83,9	87,0	95,5	85,9	72,8	91,4	89,1
700	90,3	82,9	87,3	95,8	85,9	73,3	91,1	89,0

800	91,0	83,4	86,3	96,0	85,9	73,4	90,9	89,1
900	90,3	83,5	85,0	96,6	86,0	78,1	90,5	88,8
1000	90,9	82,1	78,9	96,5	85,9	75,9	90,5	87,1
1100	90,6	81,7	73,3	96,9	85,8	74,5	90,4	85,6
1200	90,5	81,1	69,9	97,1	85,6	73,6	90,1	84,6
1300	91,0	81,3	65,5	97,4	85,6	67,9	89,8	83,8
1400	89,9	80,8	64,4	97,3	85,6	65,9	89,9	83,1
1500	89,8	82,1	61,6	97,4	85,6	64,6	89,9	82,7
1600	90,2	80,8	54,7	97,3	85,6	58,6	89,8	80,7
1700	89,8	80,7	52,8	97,3	85,6	51,5	89,6	80,2
1800	90,3	81,0	49,3	97,6	85,6	48,8	89,6	79,5
1900	89,9	81,6	49,8	97,6	85,6	46,8	89,5	79,7
2000	89,9	81,0	46,3	97,5	85,6	25,0	89,5	78,7

En la Figura 2 se grafican los valores antes representados en la

Tabla para visualizar mejor, cómo el comportamiento de la mayoría de los algoritmos se mantiene estable a partir de cierta cantidad de términos sin grandes incrementos en la exactitud predictiva. Sin embargo, en los casos de los algoritmos KNN y LVQ3 y en menor medida del LVQ2.1 se empeora el resultado con el incremento de la cantidad de términos en la colección. En la Figura 2 se ha delimitado la región de mejores resultados (valores muy próximos a 0.90) con dos líneas discontinuas verticales de color azul. La línea roja continua representa la gráfica de comportamiento del promedio entre todos los clasificadores. Se puede notar que a partir de 600 términos se alcanza el valor máximo, el cual se mantiene relativamente estable en la región delimitada, para empeorar luego producto del efecto de los algoritmos KNN, LVQ3 y LVQ2.1. El resto de los algoritmos logran un comportamiento con tendencia a mejorar o mantenerse estables con el aumento del número de términos.

Revista Cubana de Ciencias Informáticas
Vol. 10, No. 4, Octubre-Diciembre, 2016
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 154-170

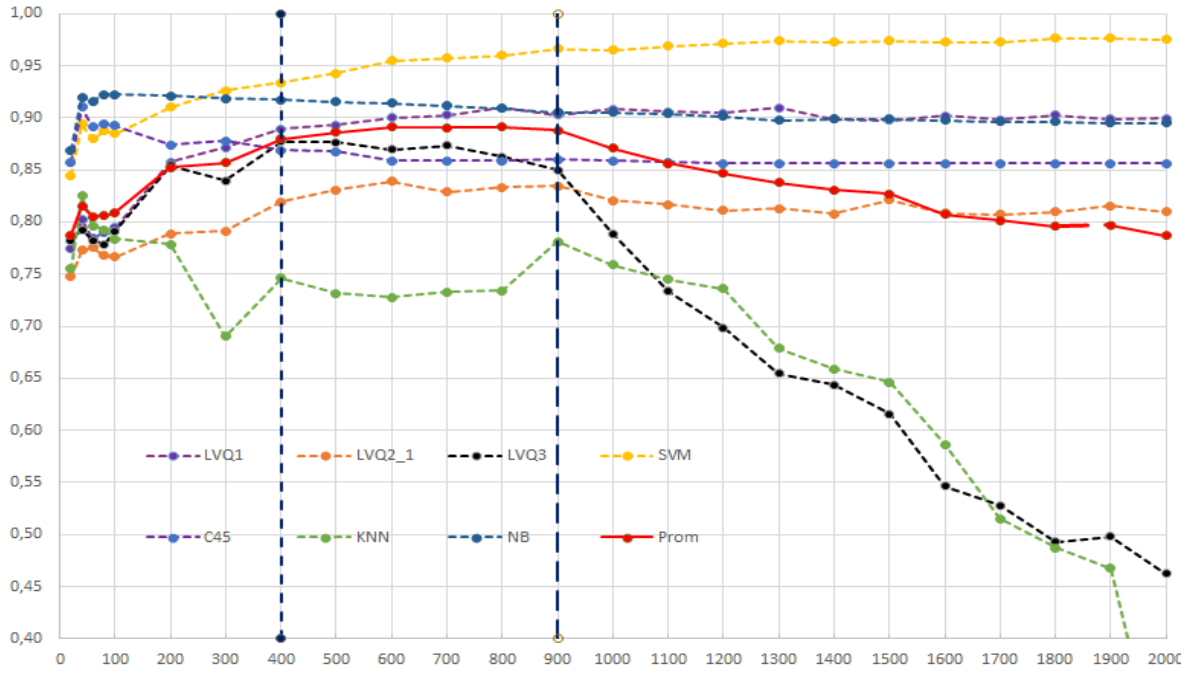

Figura 2 Gráfica del comportamiento de los algoritmos LVQ1, LVQ2.1, LVQ3, SVM, C45, KNN y NB en la colección de noticias variando la cantidad de términos entre 20 y 2000.

Conclusiones

En este trabajo se evaluó la competitividad del algoritmo LVQ en las tres versiones que se encuentran implementadas en la herramienta KEEL, fue utilizada la exactitud predictiva como métrica de evaluación.

Como tarea preliminar a este trabajo se logró crear una colección de 800 documentos textuales partiendo de noticias de medios de prensa en el idioma inglés. Utilizando los pesos $tf * idf$ del Modelo de Espacio Vectorial de Salton (Salton et al. 1975).

Se pudo acotar una región aceptable en cuanto al número de términos de la colección de noticias creada, puesto que se comprobó experimentalmente que los algoritmos SVM, NB, C45 y LVQ1 de clasificación funcionan bien a partir de solamente 400 términos de 21731 iniciales. Lográndose encontrar que alrededor de los 600 términos se encuentra un punto de muy buen desempeño para todos los algoritmos probados.

Por otro lado, aunque se encontró que los algoritmos LVQ2.1, LVQ3 y KNN funcionan bien en la región desde los 400 hasta los 900 términos se pudo notar un deterioro gradual en el desempeño de estos con el incremento de la cantidad de términos por encima del límite superior de la región anteriormente delimitada.

Como parte de este trabajo se vislumbra que puede seguir profundizándose en la evaluación de las distintas versiones del LVQ, en particular puede extenderse la implementación del KEEL para incorporar la versión OLVQ1 y así poder evaluar su desempeño en este contexto. Tomando como punto de partida la posibilidad que brinda OLVQ1 de variar la tasa de aprendizaje durante el proceso de entrenamiento. Por otro lado, puede comprobarse el desempeño del LVQ en colecciones de datos ruidosas o con valores omitidos, las cuales están disponibles en la colección que brinda la herramienta KEEL.

Referencias

AHN, K.K. & NGUYEN, H.T.C., 2007. Intelligent switching control of a pneumatic muscle robot arm using learning vector quantization neural network. *Mechatronics*, 17(4), pp.255–262.

ALCALÁ-FDEZ, J. et al., 2008. KEEL : a software tool to assess evolutionary algorithms for data mining problems. *Soft Computing*, 13(3), pp.307–318.

ALCALÁ-FDEZ, J. et al., 2011. KEEL Data-Mining Software Tool: Data Set Repository and Integration of Algorithms and Experimental Analysis Framework. *Journal of Multiple-Valued Logic and Soft Computing*, 17(2-3), pp.255–287.

ANAGNOSTOPOULOS, C. et al., 2001. Training a learning vector quantization network for biomedical classification. In *Proceedings of the international joint conference on neural networks*. National Technical University of Athens (NTUA), pp. 2506–2511.

BASHYAL, S. & VENAYAGAMOORTHY, G.K., 2008. Recognition of facial expressions using gabor wavelets and learning vector quantization. *Engineering Applications of Artificial Intelligence*, 21(7), pp.1056–1064.

BASSIUNY, A., LI, X. & Du, R., 2007. Fault diagnosis of stamping process based on empirical mode decomposition and learning vector quantization. *Int J Mach Tools Manuf*, 47(15), pp.2298–2306.

BAYES, T., 1763. An Essay towards solving a Problem in the Doctrine of Chances. *Philosophical Transactions of the Royal Society of London*, 53, pp.370–418.

BEZDEK, J.C. & Pal, N.R., 1995. Two soft relatives of learning vector. *Neural Networks*, 8(5), pp.729–743.

BLUME, M. & BALLARD, D.R., 2007. Image annotation based on learning vector quantization and localized Haar wavelet transform features. In S. K. Rogers, ed. *Society of photo-optical instrumentation engineers*. pp. 181–190.

CHEN, C.Y., 2012. Accelerometer-based hand gesture recognition using fuzzy learning vector quantization. *Adv Sci Lett*, 9(1), pp.38–44.

CORTES, C., VAPNIK, V. & SAITTA, L., 1995. Support-Vector Networks. *Machine Learning*, 20, pp.273–297.

COVER, T.M., HART, P.E. & STEVENS, K.N., 1967. Nearest Neighbor. *IEEE Transactions on Information Theory*, IT-13(1), pp.21–27.

DIETERLE, F. et al., 2003. Urinary nucleosides as potential tumor markers evaluated by learning vector quantization. *Artificial Intelligence in Medicine*, 28(3), pp.265–280.

DOMINGOS, P. & PAZZANI, M., 1997. On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. *Machine Learning*, 1997(29), pp.103–130.

DUTTA, S., CHATTERJEE, A. & MUNSHI, S., 2001. Identification of ecg beats from cross-spectrum information aided learning vector quantization. *Measurement*, 44(10), pp.2020–2027.

FIX, E. & HODGES, J.L., 1951. An Important Contribution to Nonparametric Discriminant Analysis and Density Estimation. *International Statistical Review*, 3(57), pp.233–238.

HALL, M. et al., 2009. The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11(1), pp.10–18.

HAMMER, B. et al., 2011. Prototypebased classification of dissimilarity data. *Lecture Notes in Computer Science*, 7014, pp.185–197.

HAMMER, B. & VILLMANN, T., 2002. Generalized relevance learning vector quantization. *Neural Networks*, 15(8-9), pp.1059–1068.

JOHN, G. & LANGLEY, P., 1995. Estimating Continuous Distributions in Bayesian Classifiers. In *In Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, pp. 338–345.

KARAYIANNIS, N.B., 1997. A methodology for constructing fuzzy algorithms for learning vector quantization. *IEEE Transactions on Neural Networks*, 8(3), pp.505–518.

KARAYIANNIS, N.B., 1999. An axiomatic approach to soft learning vector quantization and clustering. *IEEE Transactions on Neural Networks*, 10(5), pp.1153–1165.

KARAYIANNIS, N.B. & PAI, P.I., 1996. Fuzzy algorithms for learning vector quantization. *IEEE Transactions on Neural Networks*, 7(5), pp.1196–1211.

KOHONEN, T., 1988. An introduction to neural computing. *Neural networks : the official journal of the International Neural Network Society*, (1), pp.3–16.

KOHONEN, T., 2001. *Self-Organizing Maps* Third Exte. Springer-Verlag, ed., Berlin, Heidelberg.

KOHONEN, T., 1990. The self organizing map. *Proceeding of the IEEE*, 78(9), pp.1464–1480.

LIU, H. et al., 2002. Discretization: An Enabling Technique. *Data Mining and Knowledge Discovery*, 6(4), pp.393–423.

MARON, M.E., 1961. Automatic Indexing : An Experimental Inquiry. *Journal of the ACM (JACM)*, 8:3(January), pp.404–417.

MCCLELLAND, J.L., RUMELHART, D.E. & HINTON, G.E., 1987. The Appeal of Parallel Distributed Processing. In D. E. RUMELHART, J. L. MCCLELLAND, & others, eds. *Parallel Distributed Processing: Volume 1: Foundations*. Cambridge: MIT Press, pp. 3–44.

NEURAL NETWORKS RESEARCH CENTRE, 2007. Bibliography on the self-organizing map (som) and learning vector quantization (lvq). Available at: http://liinwww.ira.uka.de/bibliography/Neural/SOM.LVQ.html.

PAL, N.R., BEZDEK, J.C. & TSAO, E.K., 1993. Generalized clustering networks and kohonen's self-organizing scheme. *IEEE Transactions on Neural Networks*, 4(4), pp.549–557.

QIN, A.K. & SUGANTHAN, P., 2004. A novel kernel prototype-based learning algorithm. *Pattern Recognition*, 4, pp.621–624.

QUINLAN, J.R., 1993. *C4.5: Programs for Machine Learning*, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

QUINLAN, J.R., 1986. Induction of Decision Trees. *Machine Learning*, 1, pp.81–106.

RISSANEN, J., 1978. Modeling by shortest data description. *Automatica*, 14(5), pp.465–471.

SALTON, G., WONG, A. & YANG, C.S., 1975. Vector Space Model for Automatic Indexing. Information Retrieval and Language Processing. *Communications of the ACM*, 18(11), pp.613–620.

SCHNEIDER, P., BIEHL, M. & HAMMER, B., 2009. Distance learning in discriminative vector quantization. *Neural Computation*, 21(10), pp.2942–2969.

STANFILL, C. & WALTZ, D., 1986. Instance-based Learning Algorithms. *Communications of the ACM*, 12, pp.1213–1228.

WANG, J., NESKOVIC, P. & COOPER, L.N., 2007. Improving nearest neighbor rule with a simple adaptive distance measure. *Pattern Recognition Letters*, 28, pp.207–213.

WILSON, D.R. & Martinez, T.R., 2000. Reduction Tecniques For Instance-Based Learning Algorithms. *Machine Learning*, 38:3, pp.257–286.

WU, X. et al., 2007. Top 10 algorithms in data mining. *Knowledge Information Systems*, (2008)(14), pp.1–37.