Mapa Auto-organizado y Perceptrón Multicapa

Sergio García Prado

November 2, 2016

Abstract

El trabajo consiste en la implementación de un Mapa Auto-organizado en el lenguaje Matlab/Octave, el cuál se ha utilizado como "filtro" de entrada hacia el Perceptrón Multicapa que implementa la biblioteca destinada a Minería de Datos WEKA. Además, se ha realizado una comparativa sobre la tasa de acierto del SOM, del SOM + MLP y del MLP.

I. Introducción

I. Mapa Auto-organizado

Un mapa auto-organizado es un tipo de red neuronal artificial, que es entrenada usando aprendizaje no supervisado para producir una representación discreta del espacio de las muestras de entrada, llamado mapa. Los mapas auto-organizados son diferentes de otras redes neuronales artificiales, en el sentido que estos usan una función de vecindad para preservar las propiedades topológicas del espacio de entrada. [1]

El modelo fue descrito por primera vez como una red neuronal artificial por el profesor finlandés Teuvo Kohonen, debido a lo cual en ocasiones son llamadas redes o mapas de Kohonen. Al igual que la mayoría de las redes neuronales artificiales, los SOMs operan en dos modos: entrenamiento y mapeo. En el entrenamiento construye el mapa usando ejemplos entrenantes, mientras que en el mapeo clasifica una nueva entrada. [1]

Un mapa auto-organizado consiste en un conjunto de componentes llamadas nodos o neuronas. Asociado con cada neurona hay un vector de pesos, de la misma dimensión de los vectores de entrada, y una posición en el mapa. La configuración usual de las neuronas es un espacio regular de dos dimensiones, en una rejilla hexagonal o rectangular. Los mapas auto-organizados describen un mapeo de un espacio de mayor dimensión a uno de menor dimensión. El procedimiento para ubicar un vector del espacio de los datos en el mapa es encontrar la neurona con el vector de pesos más cercano (menor distancia métrica) al vector del espacio de los datos. El proceso de aprendizaje se lleva a cabo a través de la relación de vecindad entre neuronas, siendo afectadas tanto la de menor distancia a la entrada y las vecinas de la misma. [1]

II. Perceptrón Multicapa

El perceptrón multicapa es una red neuronal formada por múltiples capas, esto le permite resolver problemas que no son linealmente separables, lo cual es la principal limitación del perceptrón simple. El perceptrón multicapa puede ser total o localmente conectado. En el primer caso cada salida de una neurona de la capa "i" es entrada de todas las neuronas de la capa "i+1", mientras que en el segundo cada neurona de la capa "i" es entrada de una serie de neuronas (región) de la capa "i+1". [2]

II. Implementación

La implementación consiste en el desarrollo de un mapa auto-organizado en el lenguaje Matlab, cuya salida de la fase no supervisada se ha introducido en un perceptrón multicapa. Para ello se ha utilizado el MLP que implementa la librería WEKA. Además se ha desarrollado la fase supervisada del mapa auto-organizado, por lo que se ha podido obtener una comparativa sobre las dos alternativas.

El conjunto de datos utilizado se corresponde a una codificación numérica de manuscritos que representan números del 0 al 9. Estos han sido codificados en 40 componentes, las cuales representan un tablero de 5 x 8 casillas. Existen dos ficheros, el primero contiene 270 muestras que serán utilizadas para entrenamiento y el segundo 70 para tareas de test y comprobación de resultados.

Podemos dividir la implementación realizada en las siguientes partes según la funcionalidad que aporta cada una:

I. Mapa auto-organizado

El mapa auto-organizado implementado se ha probado con unas dimensiones de 12 x 8 neuronas, aunque este valor se puede modificar. También se han utilizado 50 épocas para la fase de aprendizaje no supervisado y un coeficiente de aprendizaje α_0 de 25. Para la fase de aprendizaje supervisado se ha realizado un etiquetado por neuronas.

Lo primero que sucede al obtener los datos de entrada es la transformación de los mismos a un formato con el que se pueda trabajar con facilidad (dado que su distribución en los ficheros no sigue una estructura que facilite la realización de operaciones). Por tanto, se divide cada uno de los ficheros en 2 matrices, que representan las muestras por filas y, los atributos por columna en el caso del primero y la clase de cada muestra en la segunda matriz (realizando la tranformación apropiada de dimension 10 original a dimensión 1 convirtiéndolo en un número $n \in [0, 9]$). Para que no surjan problemas de colisión en la fase de normalización se añade una nueva componente a todas las muestras cuyo valor es 1.

El siguiente paso es normalizar la matriz que contiene los atributos de entrada por filas para que no surjan problemas debido a la escala de los mismos. El último paso antes de comenzar con la fase no supervisada del algoritmo, se refiere a la inicialización de los pesos de las componentes de cada neurona (generados aleatoriamente en el rango $[-0.5, 0.5]$), que después son normalizados para poder aplicar en fases siguientes la propiedad del coseno para el cálculo de distancias.

- **Fase No Supervisada de SOM**: Consiste en el proceso de aproximación de los pesos de las neuronas inicializadas anteriormente hacia los valores cada componente de las muestras. Para ello, por cada muestra, se calcula la distancia a todas las neuronas del mapa para saber cuál es la más cercana. A esta neurona se la denomina *ganadora* y será en torno a la cual se produzca el aprendizaje durante dicha iteracción. Lo siguiente es calcular el conjunto de neuronas que serán adaptadas (radio de vecindad). En este caso se ha decido utilzar una estructura cuadrada cuyo lado inicial es igual al lado de menor longitud del mapa. Este radio va decreciendo con el número de iteracciones hasta que solo se ve afectada la neurona ganadora. Seguidamente, se utiliza el coeficiente de aprendizaje (que depende de la época y es carácter decreciente), para tratar de "orientar" la dirección del vector de pesos de la neurona hacia la muestra correspondiente. En este caso se ha utilizado $\alpha(t) = \frac{\alpha_0}{1+t/P}$ siendo t la época actual y P el tamaño de muestras de la misma. Este proceso se realiza tantas veces como épocas se hayan fijado (50 por defecto en esta implementación).

- **Fase Supervisada de SOM**: La fase de aprendizaje supervisado consiste en el proceso de etiquetado de clases a cada neurona, ya que lo que se ha realizado hasta el momento es el cálculo para tratar de asemejar las neuronas del mapa a la entrada. Existen dos alternativas a la hora de realizar este proceso: etiquetado global y etiquetado por neuronas. En esta implementación se ha escogido el **etiquetado por neuronas** ya que aprovecha al máximo los calculos anteriores, haciendo uso de todas las neuronas (en el etiquetado global no se utilizan todas las disponibles por no ser ganadoras para ninguna muestra). Este proceso consiste en el cálculo de las distancias de todas las entradas a cada una de las neuronas etiquetandolas con la clase de la muestra más cercana a cada una de ellas.

La implementación del mapa auto-organizado en Matlab/Octave produce lo siguiente:

- Ficheros de entrada (entrena y test) junto con las clases correspondientes en formato CSV
- Ficheros de distancias (entrena y test) a cada una de las neuronas del SOM con el filtro n^4 para facilitar la tarea de clasificación al MLP junto con las clases a las que pertenece cada muestra en formato CSV
- Estado del mapa (etiquetado de las neuronas) tras la fase de entrenamiento supervisada por salida estándar.
- Tasa de aciertos del SOM con el fichero de test por salida estándar.

II. Perceptrón Multicapa

En cuanto a la implementación del perceptrón multicapa, no se ha desarrollado por completo, sino que se ha utilizado la biblioteca WEKA, que ya posee una implementación de la misma. Por tanto, tan solo ha sido necesaria la preparación de los datos de entrada así como el ajuste de parámetros de la misma.

Para realizar la comparativa entre el uso de un Perceptrón Multicapa y la combinación del mismo con un Mapa Auto-organizado como filtro de entrada, se han utilizado los 4 ficheros anteriormente producidos por la implementación en Matlab/Octave. Para automatizar el proceso se ha creado un fichero Java en el cual se define todo el proceso para no tener que hacer uso del entorno gráfico si se quiere repetir las pruebas.

III. Shell Script

La simulación se puede llevar a cabo ejecutando el fichero shell llamado **script.sh**, que realiza la ejecución de todas las implementaciones para después poder comparar resultados.

III. Resultados

Como se puede ver en la figura 1, los resultados obtenidos únicamente con el mapa auto-organizado son los peores, a pesar de ello son los que menor coste computacional producen. En segundo lugar en cuanto a tasa de aciertos queda el Perceptrón Multicapa con la implementación en Weka. Cuando se utilizan las dos técnicas conjuntamente (utilizando el mapa auto-organizado como filtro para la entrada al perceptrón multicapa el resultado mejora). A pesar de ello, esta es la alternativa que mayor coste computacional produce.

References

[1] Wikipedia: Self-organizing map. `https://en.wikipedia.org/wiki/Self-organizing_map`

[2] Wikipedia: Multilayer perceptron. `https://en.wikipedia.org/wiki/Multilayer_perceptron`

```
--------------------------------------------------------------------------------
Title: SOM + MLP implementation comparative
Subject: Data Mining
Author: Sergio García Prado (garciparedes.me)
--------------------------------------------------------------------------------
Octave:

labels =

   1   1   1   9   0   3   2   5   1   1   1   1
   4   4   1   4   6   9   7   0   4   1   1   8
   8   7   3   3   6   2   2   2   6   9   9   5
   1   9   1   1   7   8   5   3   7   1   1   2
   8   5   9   1   5   6   2   7   2   1   1   1
   4   1   7   1   3   2   2   4   8   1   1   1
   1   1   1   3   5   8   2   2   8   1   1   9
   1   3   1   6   7   3   6   4   0   4   1   8

successRate =  0.84286
--------------------------------------------------------------------------------



--------------------------------------------------------------------------------
Weka:

SOM + MLP

Correctly Classified Instances          64               91.4286 %
Incorrectly Classified Instances         6                8.5714 %
Kappa statistic                          0.9048
Mean absolute error                      0.0221
Root mean squared error                  0.1161
Relative absolute error                 12.2766 %
Root relative squared error             38.6913 %
Total Number of Instances               70

MLP

Correctly Classified Instances          62               88.5714 %
Incorrectly Classified Instances         8               11.4286 %
Kappa statistic                          0.873
Mean absolute error                      0.0261
Root mean squared error                  0.1207
Relative absolute error                 14.4722 %
Root relative squared error             40.2171 %
Total Number of Instances               70

--------------------------------------------------------------------------------
```

Figure 1: *Resultados de las distintas implementaciones.*