Trabajo Práctico 2: Redes Neuronales Artificiales

L. Alvarez, P. Bordon y D. Santos

31 de octubre de 2016

Índice

1. Descripción del problema

Se cuenta con documentos de descripción de empresas. Dichos documentos se encuentran preprocesados, de manera tal que se tiene un dataset que especifica qué cantidad de veces aparece cada palabra en cada documento. Este formato es conocido como Bolsa de Palabras (Bag-Of-Words). El problema a resolver será diseñar y entrenar una red neuronal mediante métodos de aprendizaje hebbiano no supervisado para que pueda clasificar automáticamente los documentos según categoría. También se busca que esta red pueda generalizar corréctamente y responda de manera certera a nuevos documentos.

Se utilizaron dos subparadigmas clásicos de aprendizaje no supervisado: aprendizaje hebbiano y aprendizaje competitivo. Se trabajó en un modelo de reducción de dimensiones y otro de mapeo de características autoorganizado para su aplicación sobre las datos propuestos. En el primer caso se planteó una reducción a tres dimensiones utilizando los algoritmos de Oja y Sanger. En el segundo se planteó un modelo de autoorganización a partir del algoritmo de Kohonen. Distintas configuraciones de parámetros fueron estudiadas en cada caso. Se presentan aquí los resultados obtenidos.

2. Ejercicio 1: Reducción de dimensiones

2.1. Introducción

Para el primer ejercicio, queremos reducir la dimensionalidad de la entrada, debido a que es muy grande, a 3 valores. Para esto vamos a diseñar una red neuronal y la entrenaremos mediante aprendizaje hebbiano no supervisado aplicando la regla de oja y la regla de sanger. Luego analizaremos y compararemos sus resultados.

2.2. Análisis de la Red

En un comienzo, trabajamos con una red que contaba con 10 o 15 neuronas de salida para no reducir drásticamente su dimensión. Al final el entrenamiento, reducíamos la salida a 3 valores y obteníamos lo buscado. Luego de varias pruebas y cambios determinamos que no ganábamos nada tomando esta determinación, por el contrario, se áumentaba la complejidad del algoritmo y se perdía tiempo revisando la ortogonalidad de las 10 columnas de la matriz de pesos, mientras que los resultados obtenidos no eran mejores. Por esto se determinó manejar desde un inicio 3 dimensiones de salida.

Vale aclarar que las reglas de oja y sanger son formas de calcular el cambio que se debe hacer a la matriz de pesos para entrenarla. Por esto, ambos modelos son iguales, solo que se aplican distintos cálculos para cada tipo de regla.

2.3. Procesamiento de Datos

Sabemos que los valores de entrada son enteros positivos, ya que determinan la cantidad de apariciones de una palabra en un texto. Y al estar la entrada preprocesada y sin las palabras mas comunes (con mas apariciones), los valores de entrada se encuentran acotados. Podríamos no haber implementado ningún tipo de preprocesamiento, pero preferimos dividir los valores en 10, para que la matriz de pesos no aumente tan rápidamente y

2.4. Arquitectura definitiva

2.5. Implementación de la Red

2.6. Experimentación y Resultados

2.7. Concluciones

3. Ejercicio 2: Mapeo de características

3.1. Introducción

En este caso el objetivo fue realizar un mapa autoorganizado mediante el método de Kohonen para el mismo set de datos que en la sección anterior, con el objetivo de clasificar automáticamente los documentos de entrada en un arreglo de dos dimensiones. Para esto se probaron mapas de distinto tamaños, distinto número de iteraciones, tasas de aprendizaje y funciones de vecindad. Para la tasa de aprendizaje se consideró una función lineal monótonamente decreciente (ec 1) y una exponencial decreciente (ec 2) , para la función de vecindad se consideró una función Gaussiana, con distintas dispersiones (σ) (ec 3 4).

Taza de aprendizaje

$$v_1 = v_A \cdot e^{\frac{-t}{v_E}} \tag{1}$$

$$v_1 = v_A \cdot (1 - \frac{t}{v_C}) \tag{2}$$

Función de vecindad

$$\sigma_1 = \frac{\sigma_0}{1 + (t-1) \cdot \sigma_R} \tag{3}$$

$$\sigma_2 = \sigma_0 \cdot e^{\frac{-t}{\sigma_R}} \tag{4}$$

con t número de iteraciones, σ_R v_B v_C tomando valores

Para la realización de cada mapa se tiene en cuenta la respuesta de cada neurona sobre todos los datos considerados y se asocia dicha neurona a la categoría que más actividad genera (contando número de eventos por categoría para cada neurona). Cada color en el mapa representa una categoría diferente de empresa, existiendo nueve categorías.

3.2. Ejemplos de instancias de entrenamiento (gráficos de error, datos, pesos, etc)

Resultados del entrenamiento para los distintos parametros seleccionados

3.3. Ejemplos de instancias de validación (gráficos de error, datos, pesos, etc)

Resultados de validación para las distintas instancias de entrenamiento

3.4. Conclusiones si las hubiere