Instituto Politécnico Nacional
Unidad Profesional Interdisciplinaria de Ingenierías Campus Zacatecas
Reconocimiento de Patrones
Lernmatrix

Lernmatrix de Steinbuch

Jesús Abelardo Dávila Mauricio

20 de enero de 2021

$$M = \begin{array}{c|cccccc} & x^{1\mu} & x^{2\mu} & \cdots & x^{j\mu} & \cdots & x^{n\mu} \\ y^{1\mu} & m_{11} & m_{12} & \cdots & m_{1j} & \cdots & m_{1n} \\ y^{2\mu} & m_{21} & m_{22} & \cdots & m_{2j} & \cdots & m_{2n} \\ \vdots & \vdots & \vdots & & \vdots & & \vdots \\ y^{i\mu} & m_{i1} & m_{i2} & \cdots & m_{ij} & \cdots & m_{in} \\ \vdots & \vdots & \vdots & & \vdots & & \vdots \\ y^{p\mu} & m_{p1} & m_{p2} & \cdots & m_{pj} & \cdots & m_{pn} \end{array}$$

Introducción

La Lernmatrix de Steinbuch es el primer modelo matemático de memoria asociativa de que se tiene noticia, desarrollada en 1961 por el científico alemán Karl Steinbuch, quien publicó su artículo en una revista llamada Kybernetik, y a pesar de la importancia de su modelo y las potenciales aplicaciones, el trabajo pasó casi inadvertido.

La Lernmatrix es una memoria heteroasociativa que puede funcionar como un clasificador de patrones binarios si se escogen adecuadamente los patrones de salida; es un sistema de entrada y salida que al operar acepta como entrada un patrón binario $x^u \epsilon A^n$, $A = \{0, 1\}$ y produce como salida la clase $y^u \epsilon A^p$ que le corresponde (de entre p clases diferentes), codificada ésta con un método simple, a saber para representar la clase $k \epsilon \{1, 2, ...p\}$,se asignan a las componentes del vector de salida y^u los siguientes valores: y^u_k=1 y y^u_j=0 para j=$1, 2, ...k-1, k+1, ...p$

Fase de aprendizaje

Esquema de la fase de aprendizaje al incorporar la pareja de patrones de entrenamiento (x^u,y^u) ϵ A^n A^p

$$M = \left|\begin{array}{cccccccc} & x^{1\mu} & x^{2\mu} & \cdots & x^{j\mu} & \cdots & x^{n\mu} \\ y^{1\mu} & m_{11} & m_{12} & \cdots & m_{1j} & \cdots & m_{1n} \\ y^{2\mu} & m_{21} & m_{22} & \cdots & m_{2j} & \cdots & m_{2n} \\ \vdots & \vdots & \vdots & & \vdots & & \vdots \\ y^{i\mu} & m_{i1} & m_{i2} & \cdots & m_{ij} & \cdots & m_{in} \\ \vdots & \vdots & \vdots & & \vdots & & \vdots \\ y^{p\mu} & m_{p1} & m_{p2} & \cdots & m_{pj} & \cdots & m_{pn} \end{array}\right|$$

Cada uno de los componentes m_{ij} de M, la Lernmatrix de Steinbuch, tiene valor cero al inicio, y se actualiza de acuerdo con la regla $m_{ij} + \Delta m_{ij}$ donde:

$$\Delta m_{ij} = \begin{cases} +\varepsilon \ si \ {x_i}^\mu = 1 = {y_j}^\mu \\ -\varepsilon \ si \ {x_i}^\mu = 0 \ \mathbf{y} \ {y_j}^\mu = 1 \\ 0 \ en \ otro \ caso \end{cases}$$

siendo ϵ una constante positiva escogida previamente.

Fase de recuperación

La fase de recuperación consiste en encontrar la clase a la que pertenece un vector de entrada $x^\omega \epsilon A^n$. Encontrar la clase significa obtener las coordenadas del vector $y^\omega \epsilon A^p$ que le corresponde al patrón x^ω; en virtud del método de construcción de los vectores y^ω la clase debería obtenerse sin ambigüedad.

La i-ésima coordenada y^ω_i del vector de clase $y^\omega \epsilon A^p$ se obtiene como lo indica la siguiente expresión, donde V es el operador máximo:

$${y^\omega}_i = \begin{cases} 1 \ si \ \sum_{j=1}^{n} m_{ij} . {x^\omega}_j = \vee_{h=1}^{p} [\sum_{j=1}^{n} m_{hj} . {x^\omega}_j] \\ 0 \ en \ otro \ caso \end{cases}$$

Desarrollo

Para la fase de aprendizaje se insertaron 3 patrones con las siguientes características

```
entrenamiento.add(new PatronBinario(new int[]{1,0,1,0,1},new int[]{1,0,0},new int[]{0,0,0}));
entrenamiento.add(new PatronBinario(new int[]{1,1,0,0,1},new int[]{0,1,0},new int[]{0,0,0}));
entrenamiento.add(new PatronBinario(new int[]{1,0,1,1,0},new int[]{0,0,1},new int[]{0,0,0}));
```

y mientras que para la fase de recuperación se insertaron otros 3 patrones con los siguientes datos

```
datos.add(new PatronBinario(new int[]{1,0,1,0,1},new int[]{1,0,0},new int[]{0,0,0}));
datos.add(new PatronBinario(new int[]{1,1,0,0,1},new int[]{0,1,0},new int[]{0,0,0}));
datos.add(new PatronBinario(new int[]{1,0,1,1,0},new int[]{0,0,1},new int[]{0,0,0}));
```

Y como resultado nos arrojó la LernMatrix y sus llenados

```
| 1e , -1e , 1e ,|
| 0e , 0e , 0e ,|
| 0e , 0e , 0e ,|

| 1e , -1e , 1e ,|
| 1e , 1e , -1e ,|
| 0e , 0e , 0e ,|

| 1e , -1e , 1e ,|
| 1e , 1e , -1e ,|
| 1e , -1e , 1e ,|
```

y cada uno de los vectores nos arrojó sus clases resultantes

```
| 1e , -1e , 1e ,|
| 1e , 1e , -1e ,|
| 1e , -1e , 1e ,|
* |1|
 |0|
 |1|
 |0|
 |1|
= |3|
 |1|
 |1|
= |1|
 |0|
 |0|
```

```
| 1e , -1e , 1e ,|
| 1e , 1e , -1e ,|
| 1e , -1e , 1e ,|
* |1|
 |1|
 |0|
 |0|
 |1|
= |1|
 |3|
 |-1|
= |0|
 |1|
 |0|
```

```
| 1e , -1e , 1e ,|
| 1e , 1e , -1e ,|
| 1e , -1e , 1e ,|
* |1|
 |0|
 |1|
 |1|
 |0|
= |1|
 |-1|
 |3|
= |0|
 |0|
 |1|
```

Conclusión

Durante el la implementación del código de la Lernmatrix nos enteramos de que esta memoria asociativa es muy utilizada en patrones binarios, pero también nos percatamos de que este algoritmo se puede implementar a patrones no binarios ya que en anteriores practicas; revisábamos los dataset y en algunos se pueden interpretar valores en binario y así poder interpretar estos datos y adaptarlos a la funcionalidad de este algoritmo.

Código

https://github.com/Davila1019/PatternRecognition2020.git