|  |  |
| --- | --- |
| Título | VITIATE – INTENSIVE HARDWARE OVER ARTIFICIAL NEURAL NETWORKS |
| Asunto | DOCUMENTACIÓN GENERAL |
| Código | P1\_D0 |
| Revisión | 00.00 |
| Fecha | 15/02/2021 |
|  |  |
| Cliente | INTERNO |
| Código/Ref. | VITIATE |

Distribución

|  |  |  |
| --- | --- | --- |
| Distribución | Copias | Medio |
|  |  |  |

Trazabilidad

|  |  |  |  |
| --- | --- | --- | --- |
| Edición | Fecha | Capítulos | Modificación |
| 00.00 | 15/02/2021 | Todos | Creación del documento |
|  |  |  |  |

Contenido

[Distribución 2](#_Toc64296981)

[Trazabilidad 2](#_Toc64296982)

[Contenido 3](#_Toc64296983)

[Índice de Figuras 4](#_Toc64296984)

[Fuentes de referencia 4](#_Toc64296985)

[1 Introducción 5](#_Toc64296986)

[2 Requerimientos del PROYECTO 6](#_Toc64296987)

[3 Interfaz del algoritmo del piloto central 7](#_Toc64296988)

[4 Funcionamiento del PIloto Central 8](#_Toc64296989)

[4.1 REQ-PC-01 8](#_Toc64296990)

Índice de Figuras

**No se encontraron entradas de tabla de contenido.**

Índice de Tablas

[Tab 1. Fuentes de referencia. 4](#_Toc64297000)

[Tab 2. Parámetros del algoritmo. 7](#_Toc64297001)

Fuentes de referencia

|  |  |  |  |
| --- | --- | --- | --- |
| Ref. | Código | Título del documento | Versión |
|  |  |  |  |
|  |  |  |  |

1. Fuentes de referencia.
2. Introducción
3. Requerimientos del PROYECTO

* **[REQ-PC-01]**

1. Interfaz del algoritmo

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parámetros | | | |  |
| *Nombre Variable* | ***Descripción*** | ***Tipo variable*** | ***Valor Defecto*** | ***Rango*** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

1. Parámetros del algoritmo.
2. Funcionamiento
   1. REQ-PC-01
3. CLASES PRINCIPALES
   1. Neuron

Clase núcleo del sistema. Representa una unidad neuronal.

|  |  |  |  |
| --- | --- | --- | --- |
| Variables | | | |
| *Nombre Variable* | ***Descripción*** | ***Tipo variable*** | ***Valor Defecto*** |
| *algoIndex* | Índice del algoritmo de la función de activación de la neurona | Uchar | - |
| *inputNum* | Número de entradas de la neurona | Uint | - |
| *sumatorio* | Variable auxiliar para la ejecución de la neurona | **N\_TYPE** | 0 |
| *beta* | Coeficiente independiente | **N\_TYPE** | random[**MIN\_RANGE**,**MAX\_RANGE**] |
| *coefs* | Vector de coeficientes de la neurona | vector<**N\_TYPE**> | - |

1. Variables clase Neuron
   * 1. Constructores

Se han implementado los constructores copia habituales y constructor con parámetros, pero se ha eliminado la posibilidad de llamar al constructor sin parámetros.

|  |  |  |
| --- | --- | --- |
| Constructores | | |
|  | *Parámetros* | ***Descripción*** |
| 1º | * uint **inputNum** * uint **algoIndex**) | Constructor base. Especifica el número de entradas y el algoritmo de activación. |
| 2º | * const **Neuron**& **rhs** | Constructor copia |
| 3º | * **Neuron**&& **rhs** | Constructor copia r-values |

1. Constructores clase Neuron.
   * 1. métodos

A parte del método del algoritmo de ejecución de la neurona se han implementado otros de carácter auxiliar.

|  |  |  |  |
| --- | --- | --- | --- |
| Métodos | | | |
| *Nombre* | ***Inputs*** | ***Descripción*** | ***Outputs*** |
| *Algoritmo* | * const vector<**N\_TYPE**>& **inputs** | Evalúa la neurona para las entradas dadas | **N\_TYPE** |
| *ChangeAlgo* | * uint **algoIndex** | Cambia la función de activación | void |
| *Alfa* | * **N\_TYPE** **e** | Calcula la derivada de la función de activación para el valor ***e*** | float |
| *GetInputNum* |  | Devuelve el número de entradas | uint |
| *GetCoefs* |  | Devuelve los coeficientes de la neurona | vector<**N\_TYPE**> |
| *PrintCoefs* |  | Imprime por pantalla los coeficientes de la neurona | void |

1. Métodos clase Neuron
   1. RED

Clase núcleo del sistema. Representa la red neuronal.

|  |  |  |  |
| --- | --- | --- | --- |
| Variables | | | |
| *Nombre Variable* | ***Descripción*** | ***Tipo variable*** | ***Valor Defecto*** |
| *inputNum* | Número de entradas de la RN | Uint | - |
| *layerNum* | Número de capas de la RN | Uint | - |
| *layers* | Matriz de neuronas de la RN | Vector<vector<**Neuron**>> | - |

1. Variables clase RED
   * 1. Constructores

Se han implementado los constructores copia habituales y constructor con parámetros, pero se ha eliminado la posibilidad de llamar al constructor sin parámetros.

|  |  |  |
| --- | --- | --- |
| Constructores | | |
|  | ***Parámetros*** | ***Descripción*** |
| 1º | * uint **inputNum** * uint\* **neuronsPerLayer** * uint **layerNum**) | Constructor base. Especifica el número de entradas, el número de neuronas por capa y el número de capas. |
| 2º | * const **RED**& **rhs** | Constructor copia |
| 3º | * **RED**&& **rhs** | Constructor copia r-values |

1. Constructores clase RED.
   * 1. métodos

Los métodos de la clase RED están encaminados a la ejecución del forward y el backward.

|  |  |  |  |
| --- | --- | --- | --- |
| Métodos | | | |
| *Nombre* | ***Inputs*** | ***Descripción*** | ***Outputs*** |
| *Forward* | * vector<**N\_TYPE**> **inputs** | Evalúa la RN para las entradas dadas | vector<**N\_TYPE**> |
| *Forward* | * vector<**N\_TYPE**> **inputs** * vector<vector<**N\_TYPE**>>& **e** | Evalúa la RN para las entradas dadas almacenando los valores intermedios en *e* | void |
| *Gradient* | * vector<**N\_TYPE**> **inputs** * vector<**N\_TYPE**> **s** | Calcula el gradiente para las entradas ***inputs***, y las salidas ***s****,* sobre los coeficientes de las neuronas | vector<vector<vector<**N\_TYPE**>>> |
| *BuildMatrix* | * **Matrix**& **A** * **Matrix**& **C** * uint **layer** * vector< vector<**N\_TYPE**>>& **e** | Método auxiliar del gradiente a partir de los valores intermedios calculados por ***Fordward*** (***e***). Guarda en ***C*** los coeficientes de las neuronas y en ***A*** los valores de las contribuciones de las funciones de activación a las derivadas parciales. | void |
| *PrintCoefs* |  | Imprime por pantalla los coeficientes de las neuronas | void |
| *PrintGradient* | * vector<vector< vector<**N\_TYPE**>>> & **gradient** | Imprime por pantalla los valores calculados por el gradiente. | void |

1. Métodos clase RED
2. CLASES AUXILIARES
   1. Matrix

Clase auxiliar para el tratamiento de matrices y sus operaciones.

|  |  |  |  |
| --- | --- | --- | --- |
| Variables | | | |
| *Nombre Variable* | ***Descripción*** | ***Tipo variable*** | ***Valor Defecto*** |
| *Rows* | Número de filas de la matriz | Uint | 0 |
| *Cols* | Número de columnas de la matriz | Uint | 0 |
| *matrix* | Matriz numérica r x c | Vector<vector<**N\_TYPE**>> | - |

1. Variables clase MATRIX
   * 1. Constructores

Los constructores implementados son los tres estándar, además de la sobrecarga del operador “=”. Para inicializar la matriz se debe hacer uso de los métodos ***PlaceRow()*** o ***ColVector()***.

|  |  |  |
| --- | --- | --- |
| Constructores | | |
|  | *Parámetros* | ***Descripción*** |
| 1º | * Uint **rows** = 0 | Constructor base. Si se especifica un número de filas diferente de cero se reserva el espacio. |
| 2º | * const **Matrix**& **rhs** | Constructor copia |
| 3º | * **Matrix**&& **rhs** | Constructor copia r-values |

1. Constructores clase MATRIX.
   * 1. métodos

A parte de los métodos detallados en la Tab 8, se han sobrecargado los operadores “\*” y “[]”, donde el operador multiplicación aplica:

Siendo K el cuerpo definido por ***N\_TYPE***.

Y el operador corchete aplica:

Donde representa la delta de kronecker para m=i.

|  |  |  |  |
| --- | --- | --- | --- |
| Métodos | | | |
| *Nombre* | ***Inputs*** | ***Descripción*** | ***Outputs*** |
| *PlaceRow* | * const vector<**N\_TYPE**>& **row** | Introduce una nueva fila en la matriz en la última posición | void |
| *ColVector* | * const vector<**N\_TYPE**>& **col** | Introduce el vector columna | void |
| *ShowElements* |  | Muestra por pantalla los elementos de la matriz columna por columna | void |
| *ShowRC* |  | Muestra las dimensiones de la matriz por pantalla | void |

1. Métodos clase MATRIX