

APLICACIÓN WEB QUE MEDIANTE EL USO DE REDES NEURONALES
ARTIFICIALES RESUELVE PROBLEMAS DE OPTIMIZACIÓN NO LINEAL
(GerMath.JS)

ESTUDIANTE:
WEYNDER GERMAN AGUIRRE BETANCOUR

UNIVERSIDAD DE LA AMAZONIA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS
FLORENCIA-CAQUETÁ
2020

APLICACIÓN WEB QUE MEDIANTE EL USO DE REDES NEURONALES
ARTIFICIALES RESUELVE PROBLEMAS DE OPTIMIZACIÓN NO LINEAL.
(GerMath.JS)

ESTUDIANTE:
WEYNDER GERMAN AGUIRRE BETANCOUR
Estudiante, Universidad de la Amazonia – Florencia, Caquetá

Propuesta de trabajo de grado para optar el título de Ingeniero de sistemas de
la Universidad de la Amazonia

UNIVERSIDAD DE LA AMAZONIA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS
FLORENCIA-CAQUETÁ
2020

TABLA DE CONTENIDO

INTRODUCCIÓN	5
FORMULACIÓN DEL PROBLEMA.....	7
JUSTIFICACIÓN.....	10
OBJETIVOS.....	11
OBJETIVO GENERAL	11
OBJETIVOS ESPECÍFICOS	11
MARCO REFERENCIAL	12
PROBLEMAS DE OPTIMIZACIÓN NO LINEAL.....	12
OPTIMIZACION NO LINEAL IRRESTRICTA	12
Método del Gradiente	12
Método de Newton.....	13
OPTIMIZACION NO LINEAL RESTRICTA.....	13
CONDICIONES DE KARUSH KHUN-TUCKER (KKT).....	13
Condición necesaria de KKT.	13
Condición suficiente de KKT.....	14
CONDICIONES DE LAGRANGE.	14
Condición necesaria de LaGrange.	14
Condición suficiente de LaGrange.....	14
TÉCNICA DE LOS MULTIPLICADORES DE LAGRANGE.....	14
METODOLOGIAS DE DESARROLLO SOFTWARE.....	15
METODOLOGIAS DE DESARROLLO TRADICIONAL	16
METODOLOGIAS DE DESARROLLO AGIL	17
REDES NEURONALES ARTIFICIALES	20
Definición.....	21
Perceptrón	26
Perceptrón Multicapas (MLP)	26
Pesos de las Neuronas Artificiales	27
Función de entrada.....	27
Función de salida (activación)	27
Función de error de red	28
Retropropagación De Error.....	28

Resilent BackPropagation RProp	30
Aprendizaje Supervisado	31
Aprendizaje NO Supervisado.....	31
MARCO METODOLOGICO	32
CARACTERISTICAS DE LA METODOLOGIA XP	32
VENTAJAS Y DESVENTAJAS DE LA EXTREME PROGRAMMING.	33
VENTAJAS DE LA EXTREME PROGRAMMING.....	33
DESVENTAJAS DE LA EXTREME PROGRAMMING.	33
FASES DE LA METODOLOGIA EXTREME PROGRAMMING.....	33
FASE DE PLANEACION DE LA ENTREGA.....	33
FASE DE ITERACIONES	34
FASE DE PRODUCCION	34
FASE DE MANTENIMIENTO	34
FASE DE MUERTE	34
ENCAPSULAMIENTO EN FASES FUNCIONALES.	35
METODOLOGÍA DEL PROYECTO.....	35
Fase de Planeación	36
CRONOGRAMA	41
REFERENCIAS	44

INTRODUCCIÓN

Los problemas de optimización no lineal en su definición más simple buscan la utilización de métodos, metodologías y/o algoritmos que permitan minimizar o maximizar funciones objetivo. Estas pueden o no, estar sujetas a restricciones según sea el caso (Hernandez, 2019). Estos pueden abarcar desde la toma de decisiones en grandes organizaciones, hasta la optimización de los procesos y subprocesos productivos que se lleven a cabo en estas. Es de resaltar como precedente significativo el desarrollo del método Simplex en el año de 1947 por George Dantzig, quien proporciono el método para solucionar problemas lineales restringidos multivariados (Allende & Bouza, 2005), sin embargo años antes ya se habían realizado avances en cuanto a la optimización, de manera informal y artesanal. Esto debido a la llegada de la Revolución Industrial en 1880, que ocasiono que los pequeños empresarios comenzaran a implementar maquinaria y nuevas tecnologías, logrando un crecimiento en las organizaciones y con ello la necesidad de optimizar los procesos utilizando la Investigación de Operaciones **IO** (Lipicia Munguía Ulloa, 2005).

En su campo de acción fundamental la **IO** se utiliza para solucionar los problemas que surgen en la dirección y/o administración de sistemas de hombres, maquinas, materiales, dinero, en las industrias, en los negocios, en el gobierno, en la defensa entre otros. Notándose entonces que, debido a su enfoque cuantitativo apoyado por las matemáticas, proporciona soluciones eficientes a los problemas que pudieran originarse. Con la aparición de los ordenadores y el continuo avance de estos, se mejoró considerablemente la velocidad y capacidad de procesamiento de grandes cantidades de datos, facilitando así el manejo y solución de estos problemas en las organizaciones.

Tenemos entonces que la **IO** se puede aplicar a cualquier proceso cotidiano logrando modelar las variables que influyen en dicho momento y en el objetivo a analizar, logrando determinar de forma canónica si lo que se desea es minimizar o maximizar el modelo planteado (en la mayoría de casos lo que se busca es maximizar las ganancias o, minimizar los costos). En este desarrollo nos centraremos en la rama de la **IO** que abarca la solución de *Problemas no Lineales (PNL)*, como se ha señalado, estos se caracterizan por la ausencia de linealidad, ya sea en la función objetivo, en las restricciones del problema, o en ambos casos, nuestro problema central se establece en ¿cómo solucionar problemas de optimización no lineal restringida?

Para ello se utilizara el paradigma computacional de las Redes Neuronales Artificiales (**RNA**), el cual se inspira el comportamiento biológico de las neuronas del cerebro humano, desarrollando un sistema computacional conexionista, siendo la neurona la base del procesamiento (Torres, 2011), donde esta recibe las entradas de datos, luego se encarga de procesar la información. La eficiencia de la **RNA** depende del valor del error estimado en el cálculo, por lo cual, se recalcula el error

y se actualiza la red a partir de los nuevos datos de entrada hasta satisfacer el valor permitido, es entonces que se genera una salida. Sin embargo esto se ve potenciado en la medida que biológicamente las neuronas están diseñadas para funcionar en grupo (Graupe, 2007), la salida de una neurona puede ser la entrada de otra. De esta forma la complejidad de la red neuronal es proporcional al problema a resolver, notándose que el número de incógnitas de la función objetivo es el mismo número de entradas en la red, siendo la salida el valor máximo de ganancia o mínimo de pérdida buscado.

FORMULACIÓN DEL PROBLEMA

Basados en el avance de las tecnologías de la información a lo largo del tiempo, y su auge significativo en los años 80's y 90's, siendo determinante en los procesos de creación e innovación de los viajes espaciales, el desarrollo de la robótica, la *Inteligencia Artificial (IA)*, la *Investigación de Operaciones*, entre otras. Teniendo en cuenta que la *IO* se considera un campo de estudio bastante extenso, para este desarrollo haremos énfasis en su rama de *Programación no Lineal* más exactamente en los problemas de *Optimización No Lineal (ONL)*.

Observamos entonces que. Los problemas de optimización hacen parte fundamental del desarrollo en las organizaciones (Boirivant, 2009), aplicándose en diferentes campos tales como en la administración de recursos o materiales (Hidalgo & Díaz, 2010; Serna, Serna, & Ortega, 2010), la logística (Mejía & Elkin, 2007; Héctor & Briceño, 2011), en las redes de transporte (Trujillo, 2019), en optimización financiera (Jaramillo, Vera, & Barcia, 2018; Ospitia & Cortes, 2019), entre otras, debido a que desde su descubrimiento se generaron importantes y significativos avances frente a la solución óptima de estos problemas, ya que existen algoritmos, metodologías y/o métodos exactos para optimizar funciones de carácter lineal (Ploskas & Samaras, 2014). En contraste con la optimización de funciones no lineales donde no es posible establecer un método que aporte una solución óptima al problema cuando en la función objetivo o en las restricciones de la misma se carece de linealidad, siendo necesario aplicar **Métodos Heurísticos^a** de optimización (Gutierrez & Pulido, 2015), los cuales buscan mediante la utilización conjunta de diferentes alternativas brindar una aproximación de gran calidad a la solución óptima.

Teniendo en cuenta lo anterior, considerando el actual y acelerado crecimiento de las organizaciones, se considera indispensable contar con medidas preventivas y correctivas que ofrezcan soluciones eficientes a los problemas de optimización que se pudieran presentar sobre los modelos comerciales establecidos (Henao & Dumar, 2007; Sepúlveda, Avilez, & Jaramillo, 2014; Valdes, Aleaga, & Vidal, 2014), visto de otra forma, es necesario contar con una herramienta que se adapte a las necesidades que puedan surgir de manera esporádica o administrativa en torno a la toma de decisiones, la ejecución de procesos productivos, la minimización de costos de producción, o también para determinar la afectación organizacional debido a cambios en el valor de la materia prima, entre otros. Cabe resaltar que mediante el modelado matemático se pueden generar funciones objetivo o funciones de costo en las cuales intervienen las variables necesarias en los procesos mencionados anteriormente, usualmente estas funciones carecen de linealidad.

^a Método Heurístico: Adj. De la Heurística o relativo a ella. – Método Heurístico.

Dentro de los algoritmos o métodos reconocidos **Heurísticamente**^b (Menna, 2014) para resolver este tipo de problemas (ONL) encontramos los *Algoritmos Genéticos* (Wang, 2014), *Optimización por Enjambres* (Lin Li, 2014), la *Optimización Basada en Colonias de Hormigas* (Ínkaya, Kayaligil, & Özdemirel, 2014). En relación a lo anterior para este desarrollo aplicaremos el concepto de las *Redes Neuronales Artificiales (RNA)* buscando aplicar este paradigma en el desarrollo de una aplicación que garantice la correcta solución de los problemas ONL, teniendo en cuenta, tanto la función de objetivo, como las restricciones explícitas e implícitas que pudieran afectar dicha función limitando la zona factible de búsqueda.

Es de notar que las redes neuronales en Colombia ya han sido exploradas, estas se han utilizado en mayor manera para resolver problemas de clasificación, o regresión lineal, también en la predicción de eventos, como es el caso de la ciudad de Cali donde se diseñó e implementó una RNA para predecir la precipitación mensual en su cuenca hídrica principal, de esta manera se logró monitorear mediante 35 estaciones que enviaban datos en tiempo real a la Red, esta clasificaba la información y aportaba una predicción futura en base a los datos analizados, de esta manera se logró determinar mediante simulación a lo largo del tiempo los futuros cambios que debían realizarse y en qué zonas específicas reforzar los planes de contingencia para mitigar de mejor manera los riesgos en la población (Daniel David Montenegro Murillo, 2018).

Otro campo en el cual se ha desempeñado este paradigma en el país es en problemas de clasificación como se evidencia en un proyecto publicado por la universidad Javeriana de Colombia, donde se clasifican 200 empresas según sus aseguradoras de riesgos ARL el número de accidentes ocurridos, las indemnizaciones pagadas a los empleados, las fechas de entrada y retiro de los trabajadores, siendo las entradas de la red los datos existentes en FASECOLDA^c y el Sistema de Riesgos Laborales^d, en este caso se logró determinar que el 85% de las empresas elegidas aleatoriamente realizaban fraude ya sea en la vinculación o en el pago de las debidas indemnizaciones o remuneraciones (Hernández, 2017).

Sin embargo, las aplicaciones antes mencionadas de las RNA se aplican en el campo lineal, clasificando los datos o prediciendo futuros comportamientos, esto conlleva a la necesidad de generar una solución que facilite la optimización de funciones no lineales por parte de las empresas colombianas que en su expansión continua necesiten hacer uso de este paradigma para maximizar sus ganancias, minimizar sus costos u optimizar sus procesos, como por ejemplo los Problemas De

^b Heurística: Conjunto de técnicas, metodologías o métodos utilizados con la intención resolver problemas a través de la creatividad, pensamiento divergente o lateral, siendo el aprendizaje iterativo uno de los métodos más utilizados.

^c FASECOLDA: Federación de Aseguradores Colombianos, representa la actividad del sector asegurador frente a las entidades de vigilancia y control

^d Sistema de Riesgos Laborales: articula el sistema de prevención de accidentes de trabajo y enfermedades laborales

Asignación Generalizada (GAP) (Gutierrez C. S., 2019) en donde pueden encontrarse restricciones de capacidad no lineal al momento de optimizar la asignación de recursos, también se encuentran casos no lineales de optimización para los factores ambientales como lo es, la Planificación de los Recursos Hídricos (Villavicencio, Arumí, & Holzapfel, 2011) optando por maximizar el beneficio neto del agua teniendo en cuenta las restricciones físicas, medioambientales y económicas que pudieran surgir. El sector industrial hace uso igualmente de los métodos de optimización no lineal (Mibelli, 2005), definiendo el costo de producción como función objetivo, logrando mediante modelos matemáticos minimizar este valor, del mismo modo el sector agrícola Colombiano para maximizar su producción hace uso de funciones con objetivos múltiples, de las cuales es de destacar el cálculo del riesgo financiero que carecen de linealidad, al igual que las funciones inherentes al terreno de siembra y a la fabricación de fertilizantes (Sarmiento, 2018). Por tanto, es necesario implementar una Aplicación Web de fácil acceso, que ayude a las organizaciones a resolver sus problemas de optimización empresarial aportando soluciones efectivas y eficientes para la toma de decisiones por tanto debemos dar solución al siguiente interrogante.

¿Cómo resolver problemas de optimización no lineal restringida, haciendo uso del paradigma de redes neuronales artificiales?

JUSTIFICACIÓN

En vista del acelerado crecimiento de la población en Colombia según la revista Forbes (Staff, 2020) la cual pronostica basada en los datos poblacionales del Departamento Administrativo Nacional de Estadística (DANE) que para el año 2020 habrá en Colombia alrededor de 50,3 millones de habitantes, se genera una expansión en el desarrollo industrial, comercial y financiero de las organizaciones, las cuales deben estar preparadas para afrontar estos cambios, ya sea al hacer escalable su capacidad de producción para adaptarse a la demanda cambiante de la población objetivo, o realizando cambios logísticos en la manera como se fabrica el producto. Es de vital importancia contar con herramientas eficaces y eficientes que apoyen la toma de decisiones organizacionales, como la minimización de costos de producción, tiempos de atención al usuario, la maximización de las utilidades por producto, o de las ganancias netas del lote de producción.

A su vez, con el incremento poblacional, la tecnología también se ha visto en la necesidad de evolucionar de manera rápida, supliendo las necesidades que surgieron en esta expansión. De esta forma surgen los Sistemas de Información, que en su abrumante y acelerado cambio se han adaptado rápidamente a todos estos escalones de la modernización, entre ellos tenemos a Las Aplicaciones Web (Gil, 2008) los Servicios Web que para efectos de este desarrollo, nos proporcionaran el soporte estructural en el cual estará desplegada nuestra aplicación, disponible para ser consumida mediante cualquier dispositivo con conexión a internet, teniendo presente el buen manejo de software máquina a máquina que facilita una interoperabilidad asíncrona entre el cliente y el servidor, es por esto que las organizaciones comerciales han optado por migrar sus procesos, inventarios, y ventas a plataformas web que garanticen su expansión comercial, notándose que actualmente para las empresas es cada vez más importante contar con estas alternativas.

Esta investigación tiene como finalidad, desarrollar una aplicación Web, que haciendo uso del paradigma de Redes Neuronales Artificiales resuelva los problemas de optimización no lineal restricta que puedan generarse en estas organizaciones, notándose que no existe un método exacto para optimizar funciones objetivo NO lineales con restricciones, por tanto, para realizar este procedimiento haremos uso de la Heurística, combinando el *Método del Gradiente Descendente* y el algoritmo *Resilient Back Propagation* para así brindar una solución óptima y factible para su posterior análisis, apoyando toma de decisiones gerenciales. De esta manera lo que se busca es contribuir a la expansión empresarial en el país, enfocándonos principalmente en el departamento del Caquetá, siendo este desarrollo una importante herramienta que, con su debida utilización contribuirá a la población antes mencionada en la optimización de sus procesos, costos y utilidades, mejorado la calidad empresarial en la región y por consiguiente a la población en general, la cual será beneficiada por la expansión empresarial y laboral.

OBJETIVOS

OBJETIVO GENERAL

Desarrollar una Aplicación Web, que mediante el uso de Redes Neuronales Artificiales resuelva problemas de Optimización no Lineal Restricta.

OBJETIVOS ESPECÍFICOS

Determinar a partir de las referencias existentes, los métodos necesarios para resolver Problemas de Optimización No Lineales Restrictos.

Implementar una Aplicación Web de fácil acceso y uso que resuelva problemas de optimización no lineal Restricta haciendo uso de RNA.

Evaluar el desempeño de la Red Neuronal Artificial en base a los problemas de optimización más comunes conocidos en las literaturas de referencia.

MARCO REFERENCIAL

PROBLEMAS DE OPTIMIZACIÓN NO LINEAL.

La optimización de funciones busca minimizar o maximizar una función objetivo según requerimientos de cada organización, las funciones de carácter lineal poseen métodos exactos para optimizar el valor de las incógnitas y el generado por estas en la función objetivo, garantizando que el valor hallado es el máximo o mínimo (según corresponda) que pudiera encontrarse en la función, estando sujeta o no, a restricciones. Por otra parte, en las organizaciones surgen problemas de optimización que exceden el comportamiento lineal, ya sea en la función objetivo o en las restricciones de la misma, lo cual hace insuficiente la utilización de los métodos tradicionales de optimización como el Método Simplex, (Allende & Bouza, 2005) o el Método Gráfico.

Se debe optar en estos casos por métodos o algoritmos heurísticos que garanticen la fiabilidad del resultado, por lo regular estos algoritmos aproximan de manera iterativa e incremental los valores de las incógnitas hasta mejorar gradualmente el resultado, no obstante, existen otros algoritmos que utilizan valores de contrapeso para eliminar las restricciones no lineales de la ecuación con el objetivo de tratar la función como netamente lineal.

Ahora bien, separaremos mediante la existencia o no de restricciones los diferentes tipos de optimización no lineal para evidenciar sus diferencias y similitudes, su campo de acción y la utilización en cada uno de los casos, tenemos entonces:

OPTIMIZACION NO LINEAL IRRESTRICTA

Se les considera también métodos de descenso, estos métodos de optimización utilizan el Gradiente (∇) como medida de eficiencia operando hasta que este, o en su defecto el valor de parada del error lo permitan, es de aclarar que aunque ambos métodos se fundamentan en el concepto antes mencionado, la forma como se abordan y recorren las iteraciones hasta el resultado demarca diferencias significativas. El método del gradiente al realizar iteraciones sucesivas refleja un comportamiento ortogonal, este a su vez genera un efecto de zig-zag mientras se acerca al óptimo, por otra parte, el método de Newton posee un caso especial al solucionar funciones del tipo cuadrático ya que estas requieren solo de una iteración para alcanzar el óptimo, sin embargo, este método también puede llegar a divergir si su primera iteración se sitúa en un punto lejano del óptimo global. (Gely, 2009)

Método del Gradiente

Este método se enfoca en establecer la dirección máxima de ascenso basándose en el valor del gradiente resultante al derivar dicha función, de igual manera el máximo descenso se obtiene gracias a $(-)$ el gradiente, siendo así este método puede iterar progresivamente en busca de un mínimo o un máximo según se ajuste a las necesidades de la *Función Objetivo*.

$$\vec{X}_{k+1} = \vec{X}_k - \lambda_k * \nabla f(\vec{X}_k) \quad (1)$$

Método de Newton

Este método utiliza la segunda derivada de la función objetivo para trazar la dirección en la que se desplazará y aproximará al resultado óptimo, se fundamenta principalmente en aplicar *Series de Taylor* de 2do orden en la función objetivo, luego se minimiza esta aproximación igualando su gradiente a cero, haciendo recursivo el método, de esta manera el grado de error permitido determinará si se realizan o no más iteraciones.

$$\vec{X}_{k+1} = \vec{X}_k - \{H_f(\vec{X}_k)\}^{-1} * \nabla f(\vec{X}_k) \quad (2)$$

OPTIMIZACION NO LINEAL RESTRICTA

Al establecer restricciones funcionales, explícitas o implícitas en los problemas de optimización no lineal, se desestima el hallazgo de algún mínimo local cuyo gradiente sea igual a cero, ($\nabla f(\vec{X}_k) = 0$) ya que el punto hallado pudiera estar o no, fuera de la frontera de decisión, ahora bien si su resultado es óptimo respecto a las variables, este sería inviable respecto a las restricciones, por tanto se debe migrar de manera ágil a otros mínimos locales que cumplan a cabalidad con las restricciones de la función objetivo. Teniendo en cuenta lo anterior se han desarrollado métodos de optimización no lineal pertenecientes a las corrientes clásicas de la Investigación de Operaciones, estas tienden a transformar en lineales las restricciones para así poder aplicar los métodos clásicos de optimización lineal.

CONDICIONES DE KARUSH KHUN-TUCKER (KKT).

Condición necesaria de KKT.

Para que un punto x^* de la función sea considerado un mínimo local viable se debe cumplir la siguiente condición:

$$\begin{aligned} & \min f(x) \\ & \text{s. a } g_i(x) \leq 0 \quad i = 1 \dots n \end{aligned} \quad (3)$$

Se deben generar igual número de variables de holgura (μ) que de restricciones de desigualdad para ejercer contrapeso sobre estas y neutralizar su efecto sobre la función de objetivo de esta forma:

$$\nabla f(x^*) + \sum_{i=1}^n u_i * \nabla g(x^*) = 0 \quad (4)$$

$$\mu_i * g_i(x^*) = 0 \quad \forall i = 1 \dots n \quad (5)$$

Condición suficiente de KKT.

Si $f(x)$ y $g(x)$ son convexas, entonces podemos afirmar que el punto que cumpla con esta condición será el mínimo global de la función.

CONDICIONES DE LAGRANGE.

Condición necesaria de LaGrange.

Se aplica en casos donde las restricciones son de igualdad.

$$\begin{aligned} \min f(x) \\ \text{s. a } h_i(x) = 0 \quad i = 1 \dots n \end{aligned} \quad (6)$$

debe cumplirse estrictamente que existan $\lambda_1 \dots \lambda_n$ irrestrictos en signo tales que:

$$\nabla f(x^*) + \sum_{i=1}^n \lambda_i * \nabla h_i(x^*) = 0 \quad (7)$$

Condición suficiente de LaGrange.

Si $f(x)$ es convexa y $h_i(x)$ son lineales, entonces el método de LaGrange es suficiente. Para encontrar el mínimo global de la función.

TÉCNICA DE LOS MULTIPLICADORES DE LAGRANGE.

Considérese:

$$\begin{aligned} \min f(x) \quad (8) \\ \text{s. a} \\ h_i(x) = 0 \quad i = 1 \dots n \\ g_i(x) \leq 0 \quad j = 1 \dots n \end{aligned}$$

$$L = f(x) + \sum_{i=1}^n \mu_i * g_i(x) + \sum_{j=1}^n \lambda_j * h_j(x) \quad (9)$$

Y resolver el sistema de ecuaciones resultante:

$$\begin{aligned} \frac{dL}{dx} = 0 \quad \frac{dL}{d\mu} \leq 0 \quad \forall i \\ \frac{dL}{d\lambda} = 0 \quad \forall j \quad (10) \\ \mu_i \geq 0 \quad \forall i \quad \mu_i g_i = 0 \quad \forall i \end{aligned}$$

Tabla 1.

Resumen de los diferentes métodos de optimización no lineal

Problema	Condición Necesaria	Condición Suficiente	Técnica
$\min f(x)$	$(\nabla f(\vec{X}_k) = 0)$	f convexa	Método del Gradiente
			Método de Newton
$\min f(x)$ $s. a \ g_i(x) \leq 0$	$\nabla f(x^*) + \sum_{i=1}^n u_i * \nabla g_i(x^*) = 0$ $\mu_i * g_i(x^*) = 0$	f convexa g_i convexa	LaGrange KKT
			Multiplicadores de LaGrange
$\min f(x)$ $s. a \ g_i(x) = 0$	$\nabla f(x^*) + \sum_{i=1}^n \lambda_i * \nabla h_i(x^*) = 0$	f convexa h_j Lineales	LaGrange KKT
			Multiplicadores de LaGrange

Fuente: Adaptado del artículo *Optimización de Funciones No lineales* de la Facultad de Ciencias Físicas y Matemáticas de Universidad de Chile. (Goic, 2005)

METODOLOGIAS DE DESARROLLO SOFTWARE.

Si hablamos de la metodología de un proyecto tradicional, se basará en planear a cabalidad los detalles rigurosos que pudieran afectar en el desarrollo de las actividades o fases, sin embargo esto tiende a convertirse en un problema en el momento en que surjan cambios específicos, correcciones en los requerimientos, o al modificar el enfoque del proyecto que se esté desarrollando, sumado a esto el paradigma sistemático y pragmático de las nuevas tecnologías hace necesario contar con capacidades de adaptabilidad a cambios súbitos o esporádicos que llegaren a ser requeridos, a diferencia de las metodologías de desarrollo tradicional, estas a causa de su naturaleza iterativa deben completar una iteración antes de establecer los cambios que se deben realizar, luego de esto, planificar nuevamente los costes, duración y alcance del proyecto, esto claramente retrasaría cada una de las fases siguientes del desarrollo, lo cual es catastrófico si hablamos de presupuestos reducidos para afrontar problemas en empresas pequeñas y medianas(pymes).

En respuesta a lo anterior, se han desarrollado metodologías, no tan robustas que brindan mayor libertad y ergonomía al permitir la integración de modificaciones en los requerimientos por parte del cliente sin afectar los diferentes procesos que se estén llevando a cabo. Este es el concepto de metodología ágil de desarrollo que permite la adaptación rápida a cada cambio que pretenda el cliente, debido a que él hace parte del equipo de desarrollo, participando activamente en reuniones, aportando su punto de vista en cada problema tan pronto ocurra, una solución en

conjunto en la cual intervienen tanto los desarrolladores, el diseñador y cada uno de los integrantes del equipo de desarrollo.

Lo cual al final logra que al usuario se le entregue lo que realmente busca, ya que fue el mismo quien superviso y apporto las ideas en cada fase del desarrollo, y puesta en marcha de la aplicación, garantizando aun después de su implantación su mejora constante, hasta que todas y cada una de las necesidades o historias de usuario del cliente sean satisfechas, llegando hasta la fase de muerte donde se le otorga total control al cliente y solo se brindaría soporte en caso de necesitarse. Siendo así, las metodologías que se pueden o no utilizar ya dependen de cada desarrollador o grupo de desarrolladores, pero en el campo de la programación practica se utilizan metodologías ágiles para proyectos a corto plazo, y en proyectos bastante grandes se recomiendan metodologías tradicionales, dependiendo de cada caso.

METODOLOGIAS DE DESARROLLO TRADICIONAL

Estas metodologías se prioriza rigurosamente la calidad del desarrollo del software y su correcta documentación desde el inicio del desarrollo, con el fin de generar un software más eficiente, para ello, se hace el énfasis en planificar a cabalidad el trabajo a realizar en todas y cada una de las fases organizando por tareas cronografiadas, dependiendo cada una de la otra, siendo así una cadena de eslabones donde el más débil puede arriesgar al resto. No obstante estas metodologías brindan mecanismos para blindar estos acontecimientos y que el desarrollo no se vea afectado por los cambios o modificaciones que pudieran surgir, es entonces que empieza el proceso de producción del software, la puesta en marcha de la extrema y organizada planificación definida por procesos, roles, actividades, métodos y herramientas que se pueden usar para cumplir el objetivo organizacional del desarrollo, documentando cada cambio rigurosamente mediante UML, llevando el registro del análisis, diseño, implementación, y documentación de sistemas orientados a objetos; Entre las metodologías tradicionales o “pesadas” podemos citar:

- RUP (Rational Unified Procces) (Martínez & Martínez, 2010): Esta metodología se caracteriza por ser iterativa e incremental, estar centrada en la arquitectura y basada principalmente por los casos de uso. Incluye el modelo de casos de uso, el código fuente, diagramas de secuencia o de actividades, también se especifican claramente los roles de los usuarios.
- MSF (Microsoft Solution Framework) (Arévalo & Atehortúa, 2012): Esta metodología se caracteriza por su jerarquía organizacional, notándose que desde la presentación de la propuesta de proyecto se deben cumplir algunos parámetros fundamentales para ser considerado viable o necesario, por otra parte también al ser una metodología tradicional divide sus tareas en procesos y estos en subprocesos, que deben ser documentados como soporte estructural y organizacional del desarrollo software, las fases fundamentales de esta metodología son:

- Visión: es donde el ingeniero de sistemas levanta los requerimientos del proyecto.
 - Planificación: es donde se especifica a cabalidad las tareas a realizarse, su duración e impacto.
 - Desarrollo: es la fase donde se genera como tal el producto software necesario.
 - Estabilización: en esta fase se implanta el software en la organización y se capacita al personal sobre su correcto uso.
 - Liberación: es donde se entrega la solución software al cliente, brindando soporte solo en caso de ser necesario.
- Spiral Model (Universidad Nacional Autónoma de México, 2002): En esta metodología los procesos son representados como una espiral, no hay fases fijas tales como especificación o diseño, como su nombre lo indica, se itera en espiral corrigiendo en cada giro los inconvenientes presentados en ese momento, mitigando los riesgos originados durante el proceso, se llevan a cabo 4 tareas específicas en cada giro:
- Determinar o fijar objetivos: Se identifican objetivos específicos para la fase, también se fijan los productos a obtener, ya sea requerimientos, especificaciones, manual de usuario, se especifican las restricciones y los riesgos inmersos en ese giro.
 - Hay una cosa que solo se hace una vez: planificación inicial o previa.
 - Análisis del riesgo: son identificados y se realizan actividades para reducir los riesgos clave.
 - Desarrollo, verificar y validar: Se escoge un modelo de desarrollo para el sistema, y se prueban las tareas propias de la actividad que se está desarrollando para así evitar pruebas unitarias sobrecargadas.
 - Planificar: Se revisa a cabalidad el proyecto, se evalúa el desarrollo parcial, y se toman decisiones sobre la continuidad o por el contrario la mejora en torno a la actividad que se está llevando a cabo, luego de esto se planifica la siguiente fase de la espiral

METODOLOGIAS DE DESARROLLO AGIL

Desde 2001 se fundó “*The Agile Alliance*” una organización sin ánimo de lucro dedicada a promover los conceptos relacionados con el desarrollo ágil, integrada en sí por 17 expertos en el área del desarrollo de software para entonces, de esta forma en EEUU se empezaron a dar pasos agigantados en vías de construir una sólida metodología para el desarrollo de todo tipo de aplicaciones o proyectos, para esto se creó en ese entonces y se conserva hasta ahora, **El Manifiesto Ágil**. (Beck, 2001)

Enfocados claramente en encontrar la forma de modelar, organizar, y brindar la seguridad organizativa que pudiera encontrarse en la competencia, fue así como paso a paso se tomaron partes funcionales de otras metodologías y se fue creando un consolidado de principios y reglas que se deben seguir, en las cuales se rige esta

metodología, donde priman los intereses del cliente, la adaptabilidad, y la flexibilidad a los cambios, donde la integración, factorización y encapsulamiento priman sobre temas arbitrarios como la documentación, donde se puede generarse cambios por petición del cliente, por refactoring organizacional o haciendo reingeniería a un método obsoleto véase Tabla 1.

Tabla 2.

Comparativa entre metodologías de desarrollo ágiles y tradicionales.

Metodologías Ágiles	Metodologías Tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el proyecto	Cierta resistencia a los cambios.
Impuestas internamente (por el equipo)	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos
Pocos artefactos	Más artefactos
Pocos roles	Más roles
Menos énfasis en la arquitectura de software	La arquitectura de software es esencial y se expresa mediante modelos

Fuente: Tomado del artículo *Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP)* de la *Universidad Politécnica de Valencia*. Pág. 17.

Entre las metodologías ágiles más destacadas hasta el momento se pueden nombrar:

- XP (Extreme Programming): esta metodología establece como una característica fundamental el mantener las relaciones interpersonales como factor fundamental en el desarrollo, de esta manera se valora de manera positiva el trabajo en equipo, propiciando un buen clima de trabajo con comunicación directa y fluida entre todos los participantes. XP se caracteriza por su realimentación continua entre el cliente y el equipo de desarrollo, simplificando las soluciones para que sean más fáciles de implementar en caso tal de requerirse un cambio, esta metodología se logra agrupar en tres pilares fundamentales (Beck, 2001):

- **Historias de Usuario:** Es la forma como XP identifica los requerimientos, son tarjetas de papel en las cuales el cliente plasma brevemente los requerimientos o características que el sistema deberá poseer, pueden ser requisitos funcionales o no funcionales. Es de aclarar que gracias a la flexibilidad de la metodología en cualquier momento las historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, se debe delimitar cada historia de usuario de tal manera que los programadores puedan implementarla en unas semanas
- **Roles:** Aunque en las diferentes literaturas aparecen algunas variaciones y extensiones de roles XP, describiremos los roles de acuerdo con la propuesta original de Beck
 - *Cliente:* Responsable de definir y conducir el proyecto, de igual manera los objetivos.
 - *Programadores:* Estiman el tiempo y el coste del proyecto, además de llevarlo a feliz término en cuanto a los rigores del desarrollo.
 - *Tester:* Encargado de pruebas.
 - *Tracker:* Encargado del seguimiento.
 - *Big Boss:* Gestor del proyecto.
- **Proceso:** El proceso cíclico de desarrollo software en resumen consiste en los siguientes pasos:
 - 1. El cliente define el valor de negocio a implementar.
 - 2. El programador estima el esfuerzo necesario para su implementación.
 - 3. El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.
 - 4. El programador construye ese valor de negocio.
 - 5. Vuelve al paso 1.

Teniendo en claro la flexibilidad de las fases de esta metodología, es de vital importancia resaltar que en todas las iteraciones que pudieran presentarse en un ciclo tanto el cliente como el desarrollador deben establecer las prioridades y riesgos que pudieran surgir mientras avanza el proyecto. Es por esta planificación inicial que no se debe presionar al programador a realizar más trabajo que el estimado, ya que esto supone una pérdida de calidad en el software o un retraso en los tiempos de entrega del proyecto. El ciclo de vida ideal de XP consta de seis fases: Exploración, Planificación de la Entrega (Release), Iteraciones, Producción, Mantenimiento y Muerte del Proyecto.

- DSDM (Dynamic Systems Development Method) (Molina, 2013): es una metodología de desarrollo creada para entregar la solución correcta en el

momento correcto. Utiliza un ciclo de vida iterativo, dividiendo el proyecto en periodos cortos de tiempo, definiendo entregables para cada uno de estos periodos, esta metodología cuenta con roles claramente.

Los principios esta metodología se establecen en la necesidad del negocio como eje central, las entregas a tiempo, el trabajo colaborativo, nunca comprometer la calidad del desarrollo por apresurar las entregas; desarrollar de modo incremental sobre una base sólida, aplicando el desarrollo iterativo, la comunicación clara y continua para llegar así a la demostración de control

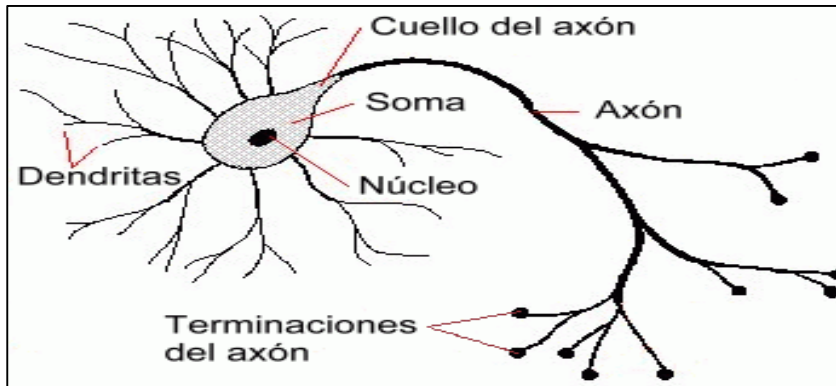
- ASD (Adaptive Software Development) (Cadavid, Martínez, & Vélez, 2013): esta metodología se fundamenta la teoría de sistemas adaptativos complejos. Es decir, interpreta los proyectos de software como sistemas adaptativos complejos compuestos por agentes, entornos y salidas, lo cual sería el proyecto terminado. El ciclo de vida de ASD está orientado al cambio y se compone de las fases: especulación, en esta fase se inicia el proyecto y se planifican las características del software, colaboración en esta fase desarrollan las características previamente identificadas determinado los tiempos y costos del desarrollo de cada componente, y aprendizaje en este punto, se revisa su calidad, y se entrega al cliente, la revisión puede hacer necesarias más iteraciones en el desarrollo del producto antes de su entrega final, también posee características propias de las metodologías del tipo ágil, como ser iterativas, tener marcos de tiempo especificados, además está orientado a los componentes software más que a las tareas y es tolerantes a los cambios.

REDES NEURONALES ARTIFICIALES

Neuronas Biológicas: La neurona es la unidad funcional y estructural del sistema nervioso, produce y transmite impulsos eléctricos, estos a su vez propagan y almacenan información para posteriormente ser usada en el razonamiento o en la toma decisiones, véase *Figura 1*. Se encuentra formada por tres partes: el cuerpo neural o soma, el axón, las dendritas y zonas de conexión entre una neurona y otra llamada, sinapsis. (Boeree, 2009)

Figura 1.

Representación gráfica de la neurona biológica.



Fuente: Tomado del libro *Atmosferas en el Capítulo VI* pág. 3-4 (Universidad de Murcia, 2014)

Definición.

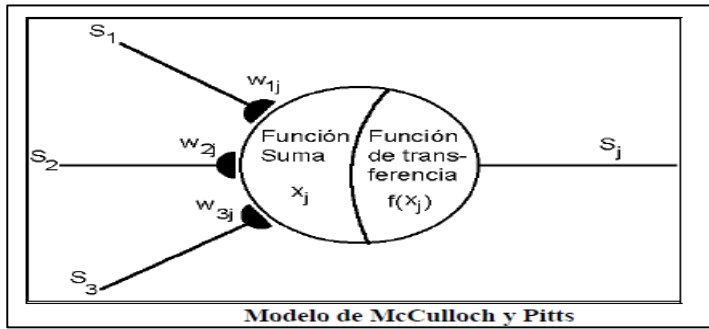
Las Redes Neuronales Artificiales (RNA) son modelos inspirados en la estructura neuronal del sistema nervioso, debido a esto, intentan replicar su comportamiento, no obstante sus métodos se pueden establecer como la abstracción computacional de una serie de características propias del cerebro humano, las RNA en su principio básico logran aprender mediante la iteración, almacenando la experiencia de eventos pasados para aplicarla a eventos futuros, las RNA pueden ser entrenadas para llevar a cabo tareas específicas, también se puede ejercer sobre ellas un entrenamiento iterativo que como explicamos anteriormente conlleva a un aprendizaje, y posterior a eso la adaptación del conocimiento en una mejora continua, es de notar que existen diferentes algoritmos de aprendizaje como son: el aprendizaje supervisado, el no supervisado y el aprendizaje por refuerzo, (Olabe, 2010) cada uno de ellos con características diferentes en cuanto a la forma de abordar el criterio de error en base a la salida deseada, siendo el aprendizaje no supervisado el que asemeja de manera más lógica el comportamiento biológico del cerebro, sin embargo ambos algoritmos de entrenamiento tienen como objetivo el correcto aprendizaje de la red neuronal en base a el conocimiento impartido.

Las RNA se componen de neuronas conectadas entre sí, estas transmiten información de forma innata, reciben, procesan y envían datos entre sí, estas neuronas interconectadas responden al aprendizaje impartido, retornando una salida con la respuesta a la tarea específica para la cual fue entrenada la red. Uno de los primeros modelos matemáticos de una neurona fue el propuesto por McCulloch y Pitts en 1.943 (Pitts & Mcculloch, 1943) y en él, se basan las redes neuronales actuales, en este modelo cada neurona consta de un conjunto de entradas S_i , y una sola salida, S_j , cada entrada i es afectada por un coeficiente denominado peso y que se representa por la letra W_{ij} . El subíndice i refleja que el

peso afecta a dicha entrada, y j hace alusión a la neurona en ese determinado momento, véase *Figura 2*.

Figura 2

Modelo de Neurona de McCulloch y Pitts



Fuente: Tomado del Artículo *Redes Neuronales Artificiales* de la Universidad Nacional de Colombia pág. 10 (Torres, 2011)

Entonces: la cantidad referente a la suma del producto de cada entrada multiplicada por su peso respectivo se denomina X_j donde j es el indicador de cada uno de los productos antes definidos, luego todos y cada uno de los valores de X_j son multiplicados por la *Función de Transferencia* $f(X_j) = S_j$ logrando así la “**Activación**” de los valores obtenidos, y generando una salida S_j

Q_j = es un valor “umbral”

$$X_j = \text{Sum } S_i W_{ij} + Q_j \rightarrow (11)$$

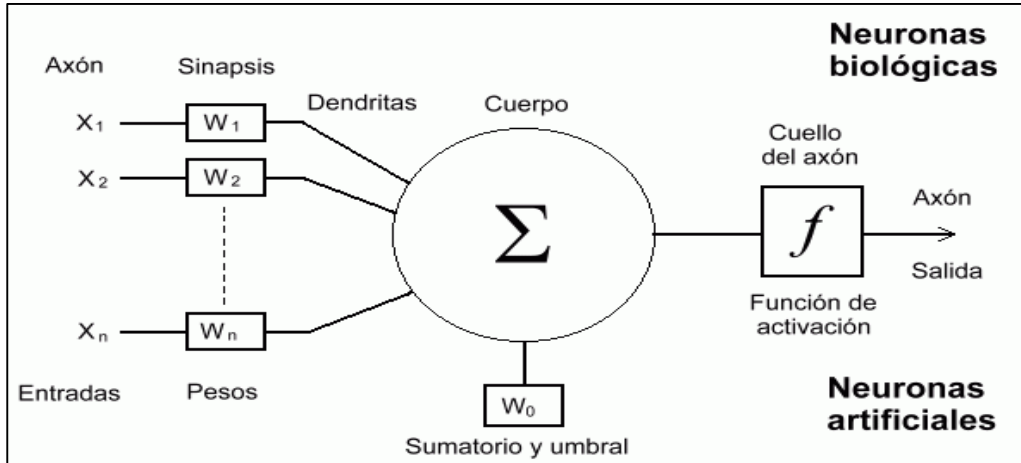
De esta forma:

$$S_j = f(X_j) (12)$$

La Figura 3, expone un ejemplo de modelo neural con n entradas, que consta de: Un conjunto de entradas X_n , los pesos sinápticos W_n, \dots, W_n , correspondientes a cada entrada, una sumatoria Σ , que almacenara los productos de $W_n X_n$, una función de activación $f()$. La cual mantiene el conjunto de valores de salida normalmente entre (0,1) o (-1,1), haciendo uso de la función sigmoidea, véase *Tabla 2*, y una salida Y .

Figura 3

Modelo de comparativo de neuronas (Biológica – Artificial).



Fuente: Tomado del Website *Redes Neuronales: una visión superficial* de la Universidad de Sevilla, (Caparrini, 2018).

Las entradas son el estímulo que la neurona ha recibido del entorno, y la salida es la respuesta a tal estímulo. La neurona puede adaptarse y aprender del medio en que se encuentre, modificando el valor de sus pesos sinápticos, por ello son conocidos como los parámetros libres del modelo, ya que pueden ser modificados y adaptados, en este modelo, la salida neuronal Y está dada por:

$$Y = f\left(\sum_{i=1}^n \omega_n x_n\right) \quad (13)$$

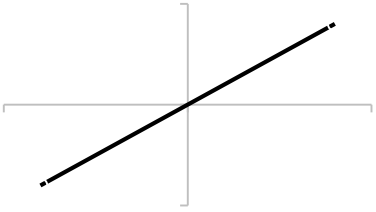
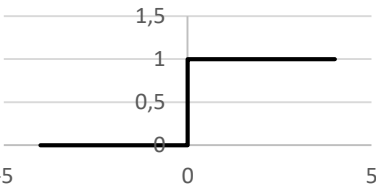
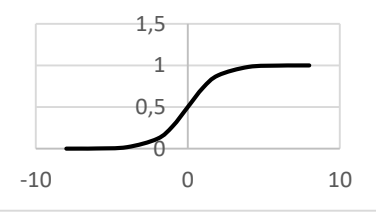
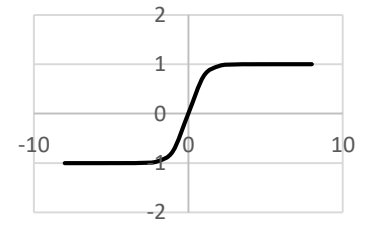
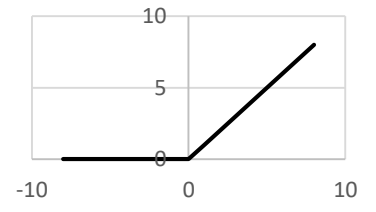
La función de activación se elige de acuerdo a la tarea realizada por la neurona, se establecen cinco tipos representativos de funciones de activación como lo son:

- *Función de Activación Lineal:* se aplica generalmente en problemas de regresión lineal, ubicándose esta función de activación en la capa de salida, teniendo un rango de valores de entrada desde $-\infty$ hasta ∞ .
- *Función de Activación Escalón:* se caracteriza por generar un valor de salida 0 para todos y cada uno de los elementos de entrada menores o iguales a 0, o tomar el valor de salida 1 para cada elemento de entrada positivo diferente de cero, esta función de activación suele encontrarse en problemas de selección y se ubican en la neurona final, logrando de esta manera tomar o no la decisión a seleccionar.

- *Función de Activación Sigmoides*: esta función de activación es ampliamente utilizada en el campo de la I.A gracias a su naturaleza no lineal que permite la agrupación y clasificación de los datos en dos categorías fundamentales, estando acotada dicha función entre los valores de salida 0 y 1, logrando estabilizarse en los valores de salida planteados entre más grande su valor de entrada. Es de aclarar que la evaluación del punto 0 en esta función equivale a 0,5 y todos los valores negativos subsecuentes a este tienden a estabilizarse en 0, al igual para los valores positivos que se evaluarán en dicha función, estos se estabilizan en 1.
- *Función de Activación Tangencial Hiperbólica*: esta función de activación surgió como solución al problema de la simetría respecto a los valores negativos presentados en la Función de Activación Sigmoides, por tanto es una función de activación no lineal, su acotamiento se encuentra en -1 y 1 logrando así generar salidas negativas en base a la función objetivo a ajustar, lo cual a su vez se potencia con el algoritmo del gradiente descendente, haciendo de esta una de las funciones de activación más utilizada en la I.A.
- *Función de Activación ReLU*: de las siglas en inglés, "Rectified Linear Unit" esta función de activación como característica principal genera un valor de salida 0 para todos y cada uno de los valores de entrada negativos, a su vez, un aumento proporcional en el valor de salida para todos los valores de entrada positivos, por lo regular se utiliza en capas profundas de las redes neuronales, esta función de activación es ampliamente usada en el reconocimiento de imágenes y voz.

Tabla 3.

Diferentes tipos de función de activación.

	Función	Rango	Gráfica
Lineal	$y = x$	$[-\infty, \infty]$	
Escalón	$f(x) = \begin{cases} 0 & \text{si } x < 0 \\ 1 & \text{si } x \geq 0 \end{cases}$	$\{0, 1\}$	
Sigmoidea	$f(x) = \frac{1}{1 + e^{-x}}$	$[0, 1]$	
Tangente Hiperbólica	$f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$	$[-1, 1]$	
ReLU	$f(x) = \max(x, 0)$	$[0, \infty]$	

Fuente: Adaptado del artículo *Máquinas de soporte vectorial y redes neuronales artificiales en la predicción del movimiento* de la revista *Odeón*, Número 9 pág. 121. (Anzola, 2015)

Perceptrón

El perceptrón es la red neuronal más básica que existe, teniendo en cuenta que su aprendizaje es supervisado, se remonta a la década de los años 1.950. El funcionamiento es muy sencillo, simplemente lee las entradas, suma todas las entradas de acuerdo a unos pesos y el resultado lo introduce en una *función de activación* que genera el resultado final, el entrenamiento del perceptrón no es más que determinar los pesos sinápticos y el umbral que haga que la entrada se ajuste a la salida. Para determinar esto, se sigue un proceso adaptativo e iterativo, el proceso da inicio con valores aleatorios y se van modificando estos valores según la diferencia entre los valores deseados y los calculados por el perceptrón utilizando la medida del error para aproximar el próximo resultado o la próxima salida. (Calvo, 2018; Blazquez, 2005)

1. Inicializar pesos y umbrales
2. Bucle: hasta resultado de pesos sea aceptable
 - Bucle: para todos los ejemplos
 - Leer valores de entrada
 - Calcular error
 - Actualizar pesos según el error
 - Actualizar pesos de entradas
 - Actualizar el umbral

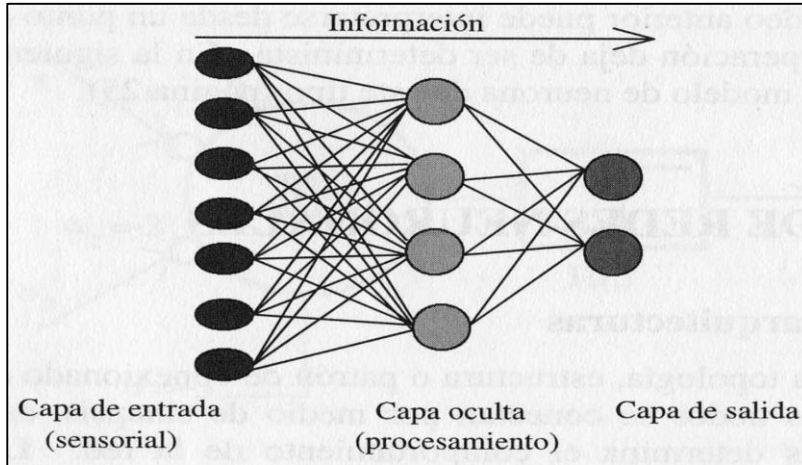
Nota: Solo es capaz de representar funciones lineales debido a que no dispone de capas ocultas.

Perceptrón Multicapas (MLP)

Es un tipo de red neuronal que tiene una capa de entrada y una capa de salida, y una o más capas ocultas en medio. Cada capa tiene un número de neuronas fijo, pero potencialmente diferente, es decir, será fijo durante todas iteraciones o épocas de la capacitación, después de obtener el resultado esperado, o que el umbral de error sea lo suficientemente pequeño para determinar que la solución es aceptable e idónea, se podrá modificar tanto la cantidad de parámetros de entrada, las épocas de capacitación o el objetivo a resolver. Cada conexión de neuronas tiene un peso, una función de entrada y una función de salida, véase Figura 4.

Figura 4.

Ejemplo Gráfico de Perceptrón Multicapa.



Fuente: Tomado del *Departamento de Ciencias de la Computacion e Inteligencia Artificial de la Universidad del País Vasco*, (Larranaga, Inza, & Moujahid, 2005)

Pesos de las Neuronas Artificiales

En una red biológica, se envía una señal eléctrica por el axón hasta las dendritas de las neuronas. La fuerza de la señal determina la cantidad de influencia que la neurona encendida tiene sobre las demás neuronas, las RNA son el análogo a este tipo de red biológica, y simula la fuerza de la señal por medio del peso de la conexión de salida, que claramente podemos definir, pero a su vez la red neuronal puede ajustar según sus necesidades. De esta forma el peso se reparte entre la capacidad de ajustarse entre épocas de capacitación, y la fuerza de la señal, que denota la influencia en la red.

Función de entrada

El tipo de función de entrada más habitual se llama *Suma Ponderada*. Una neurona de entrada n está conectada a n neuronas de la capa anterior de la red, la suma ponderada A de las entradas para n se calcula añadiendo el producto del valor de entrada de cada conexión por A veces el peso de la conexión ω a lo largo de todas las entradas n . (Larranaga, Inza, & Moujahid, 2005)

$$A_N = \sum_{i=1}^n a_i \omega_i \quad (14)$$

Función de salida (activación)

El tipo de función de activación que se usa más habitualmente en las redes MLP es la *Función de Activación Sigmoidea* antes mencionada (véase Tabla 2) , tenemos entonces que para una suma ponderada X en una neurona determinada, el valor sigmoideo V de x , es calculado por:

$$V_x = \frac{1}{1 + e^{-x}} \quad (15)$$

Función de error de red

La función de error más usada con las redes MLP es la *función de error cuadrático medio*. La cual calcula la "distancia" media entre el valor real que el programa calcula, y el valor esperado de los datos de entrenamiento, dadas n neuronas de salida, para cada suma ponderada de la salida X , el programa de capacitación calcula la diferencia entre el valor de los datos de capacitación y el valor de la red, lo eleva al cuadrado, suma esos valores de todas las neuronas de salida y lo divide por el número de neuronas de salida n para llegar al error de salida total E . (IBM, 2001)

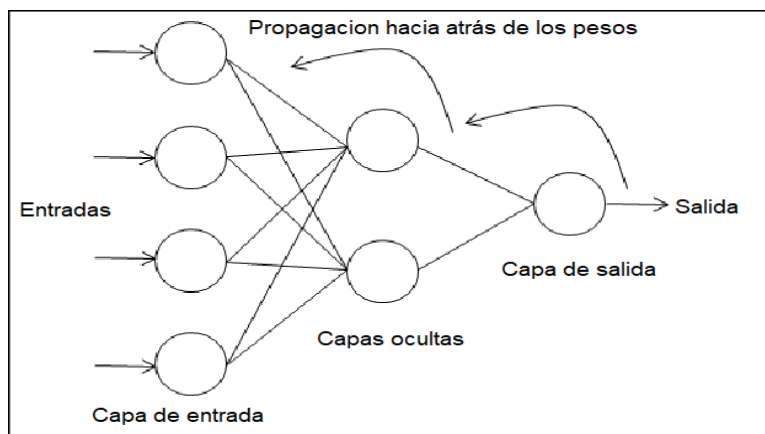
$$E_o = \frac{1}{n} \sum_{i=1}^n (A_i \text{Esperado} - A_i \text{Actual})^2 \quad (16)$$

Retropropagación De Error

También conocido como *BackPropagation* cuando una época de capacitación acaba, el programa de capacitación calcula el error de la red neuronal, y modifica los pesos de la conexión a lo largo de la red, empezando en la capa de salida y propagando el error hacia atrás, hacia la capa de entrada, ajustando el peso de cada conexión neuronal mientras recorre la red (véase Figura 5), a esto se le conoce como retropropagación de error y es una técnica ampliamente utilizada para capacitar redes neuronales.

Figura 5.

Red Neuronal Artificial, Visualización del Algoritmo BackPropagation.



Fuente: Traducido del artículo, *training a Functional Link Neural Network Using an Artificial Bee Colony for Solving a Classification Problems*, pág. 3 (Hassim & Ghazali, 2012)

Como se mencionó anteriormente, el algoritmo de la propagación hacia atrás o *BackPropagation* se utiliza con frecuencia en la capacitación de RNA aplicadas a la clasificación de datos, imágenes o también en la separación lineal de variables. Lo que hace realmente potente este algoritmo es su capacidad de adaptación gradual, propagando los datos desde las neuronas de la capa de entrada hasta la capa de salida a través de las conexiones en las capas de neuronas, al llegar a la capa de salida o a la neurona final se calcula el error entre los valores arrojados por la RNA y los esperados en el entrenamiento, entonces se hace uso nuevamente de las conexiones en las neuronas de la capa oculta para transportar los datos respectivos operándolos con el peso de cada conexión, cabe aclarar que al propagar el valor del error hacia atrás, el valor a corregir de cada neurona será proporcional a la aportación de esta en el valor de salida generado en comparación con el valor de salida esperado por la red, es entonces que se itera nuevamente con el valor los pesos actualizado, en busca de identificar los valores que satisfagan las entradas necesarias para generar la salida correcta, tenemos entonces que:

$$n_i^o = \sum_{j=1}^q W_{ij}^o X_j + b_i^o \quad (17)$$

n_i^o = Entrada neta de la neurona i a la capa oculta.

q = Corresponde a la cantidad de entradas de la RNA.

W_{ij}^o = Peso de la conexión de la entrada i con la neurona de la capa oculta j .

X_j = El valor de la entrada j a la RNA.

b_i^o = Corresponde al valor bias de ganancia de la neurona i de la capa oculta.

Nota: La notación (o) representa la capa a la que pertenece el respectivo parámetro. Ahora bien, cada neurona de la capa oculta recibe la entrada n_i^o respectiva, y a su vez genera una salida luego de su paso por la función de activación asociada a esta neurona, de esta forma se genera la siguiente ecuación:

$$a_i^o = f^o \left(\sum_{j=1}^q W_{ij}^o X_j + b_i^o \right) \quad (18)$$

Donde, f^o representa la función de activación de la capa oculta, y a_i^o como se mencionaba anteriormente si bien es la salida de la capa oculta a su vez es el valor de entrada de la capa de salida, $a_k^o \Rightarrow n_k^s$.

$$n_k^s = \sum_{j=1}^m W_{kj}^s a_j^o + b_k^s \quad (19)$$

m = Numero de neuronas de la capa oculta.

s = Numero de neuronas de la capa de salida.

W_{kj}^s = Peso de la conexión de la neurona j con la neurona k de la capa de salida.
 a_j^0 = El valor de la salida de la neurona oculta j .
 b_k^s = Corresponde al valor bias de ganancia de la neurona k de la capa de salida.
 n_k^s = Corresponde al valor de la entrada neta a la capa de salida.

$$a_k^s = f^s(n_k^s) \quad (20)$$

Donde, f^s representa la función de activación de las neuronas de la capa de salida, de esta forma la ecuación de la salida de la RNA está dada por:

$$a_k^s = f^s\left(\sum_{j=1}^m W_{kj}^s a_j^0 + b_k^s\right) \quad (21)$$

Luego esta salida será comparada con la salida esperada t_k y se calcula el error, haciendo uso de la ecuación (6) que determinara el grado de error y de ajuste que se propagara hacia atrás capa a capa, neurona a neurona en busca de la optimización del valor del error hasta un porcentaje permisible.

Resilient BackPropagation RProp

RProp es una técnica utilizada en el entrenamiento supervisado de redes neuronales artificiales el cual se basa en la búsqueda de la derivada de $f(x)$; en este caso, $f(x)$ establece una relación entre la diferencia arrojada por la salida de la RNA y el valor esperado (Riedmiller, 1994). Se diferencia del algoritmo BackPropagation principalmente por que las derivadas parciales de la función error se usan única y exclusivamente para determinar el sentido en que se deben corregir los pesos de la red, pero no la magnitud del ajuste, otra característica que las diferencia es que BackPropagation modifica el valor del paso o ajuste proporcionalmente al gradiente resultante de la función de error, no obstante si se llega a una condición donde el gradiente tiende a ser plano o muy cercano al 0 el algoritmo avanza proporcional a este valor por tanto su paso será mínimo y su eficiencia se verá afectada.

RProp por otra parte genera un único parámetro que establece la velocidad de avance, este parámetro recorre la función objetivo para todos y cada uno de los pesos de la red neuronal, logrando así un avance constante solo siendo restringido por el umbral de parada al encontrar los parámetros óptimos. Este algoritmo usa la derivada solamente para determinar la dirección en la actualización de los pesos. Gracias a esto y su bajo consumo computacional, converge de manera más eficiente que los algoritmos que se basan en BackPropagation.

Tenemos entonces:

$$W_{ij}^{(k+1)} \leftarrow W_{ij}^{(k)} + \Delta W_{ij}^{(k)} \quad (22)$$

ΔW_{ij} = Función de cambio de signo de la derivada del error respecto a las iteraciones (k) y (k+1).

Δ = Tamaño del paso. (Gradiente).

Δ_{max} = Tamaño máximo que puede llegar a tomar el paso.

Δ_{min} = Tamaño mínimo que puede llegar a tomar el paso.

Si, el signo de la derivada no cambia en las últimas dos iteraciones, se aumenta el tamaño del paso en η^+ , $\eta^+ \leq \Delta_{max} = W_{ij}$, si hay un cambio de signo en la derivada de la función se supera el punto de ajuste mínimo por tanto se debe reducir el tamaño del paso en una cantidad η^- , sin sobrepasar el factor Δ_{min} , siendo $\eta^- \geq \Delta_{min} = W_{ij}$ en caso de ser cero la derivada, no se realizan modificaciones.

Aprendizaje Supervisado

Ocurre cuando se toma un conjunto de datos muestrales de ejemplo, y cada ejemplo está formado por una entrada y una salida deseada. Se introducen entradas hasta que la red neuronal arroje un porcentaje de fiabilidad pertinente en la salida deseada.

Entonces tenemos:

1. *alimentar la red con datos conocidos.*
2. *¿La red arroja respuestas correctas?*
 - *Sí (dentro del porcentaje de fiabilidad): Ir al paso 3.*
 - *No:*
 1. *Ajustar los pesos de las conexiones de la red.*
 2. *Ir al paso 1.*
3. *Fin.*

Aprendizaje NO Supervisado

El entrenamiento no supervisado es aquel en que la red tiene que entender las entradas sin ayuda de la salida, cabe aclarar que en su gran mayoría se prefiere utilizar aprendizaje supervisado; tenemos entonces que la red es alimentada con entradas, pero no con las salidas deseadas, el sistema decidirá las características y atributos que deban tener los datos para ser agrupados, es en este punto donde es imprescindible el cálculo del error, debido a que con la utilización de métodos numéricos podemos aproximar a la salida optima en base al error obtenido, iterando hasta lograr un a medida de error aceptable.

MARCO METODOLOGICO

CARACTERISTICAS DE LA METODOLOGIA XP

Simplicidad: La simplicidad consiste en desarrollar solo el sistema que realmente se necesita. Implica resolver en cada momento solo las necesidades actuales.

Feedback: Una metodología basada en el desarrollo iterativo de pequeñas partes, con entregas y pruebas frecuentes y continuas, proporciona un flujo de retroinformación valioso para detectar los problemas o desviaciones.

Decisión:

- Implica saber tomar decisiones difíciles.
- Reparar un error cuando se detecta.
- Mejorar el código siempre tras el feedback y las sucesivas iteraciones.

Comunicación: Algunos problemas en los proyectos tienen origen en que alguien no dijo algo importante en algún momento, XP hace casi imposible la falta de comunicación, ya que pone en comunicación directa y continua a clientes y desarrolladores.

Planificación: Se hacen las historias de usuario y se planifica en qué orden se van a hacer y las mini-versiones, la planificación se revisa continuamente.

Versiones Pequeñas: Las versiones simplificadas deben ser lo suficientemente modulares como para poder hacer una en pocas semanas. Deben ser versiones que ofrezcan algo útil al usuario final y no fragmentos de código que no pueda ver funcionando.

Diseño Simple: Hacer siempre lo mínimo imprescindible de la forma más sencilla posible, mantener siempre sencillo el código.

Integración Continua: Debe tenerse siempre un ejecutable del proyecto que funcione y en cuanto se tenga una nueva pequeña funcionalidad, debe recopilarse y probarse. Es un error mantener una versión congelada dos meses mientras se hacen mejoras y luego integrarlas todas de golpe.

Ritmo Sostenible: Se debe trabajar a un ritmo que se pueda mantener indefinidamente. Esto quiere decir que no debe haber días muertos, en que no se sabe qué hacer y no se deben hacer en exceso horas otros días. Hay que trabajar para conseguir el objetivo cercano de terminar una historia de usuario o una mini-versión.

VENTAJAS Y DESVENTAJAS DE LA EXTREME PROGRAMMING.

VENTAJAS DE LA EXTREME PROGRAMMING.

Una de sus principales ventajas o la más resaltante a la hora de emprender los desarrollos es la gran adaptabilidad que ofrece esta metodología, siendo así escalable, adaptable y elástica, gracias a esto su utilidad y versatilidad se ven inmersas en lo amplio de su campo de acción, otra característica resaltable es la optimización del tiempo de desarrollo, al centrarse solo en los detalles funcionales de la aplicación, dejando de lado detalles o propiedades que no atacan agresivamente la funcionalidad del sistema(Diseño, Documentación, Soporte de Manual).

DESVENTAJAS DE LA EXTREME PROGRAMMING.

Su principal desventaja es que los costes de tiempo y costo de obra de mano, o desarrollo funcional, no pueden ser definidos al inicio del proyecto, debido a su naturaleza iterativa e incremental, que soluciona fallas tan pronto son detectadas, lo cual hace que se aumente el tiempo de desarrollo y el coste, no obstante, también es de aclarar que, en felices términos sin sobresaltos generaría un ahorro sustancial en el tiempo de desarrollo funcional de la aplicación.

FASES DE LA METODOLOGIA EXTREME PROGRAMMING.

FASE DE EXPLORACION

En esta fase se exponen a grandes rasgos las necesidades del cliente en las historias de usuario, para la primer entrega del producto, se prueba las tecnologías que serán utilizadas, y se construye un prototipo con las mismas, evaluando el desempeño y la posible respuesta ante las necesidades venideras, de aquí es donde parte la fase de planeación, sin perder de vista la individualidad de la misma ya que el cliente puede redactar más historias de usuario si lo considera necesario, se recomienda que las historias de usuario no superen las 3 líneas. (Letelier & Penadés, 2012)

FASE DE PLANEACION DE LA ENTREGA

Ya en esta fase se establece la prioridad de cada historia de usuario o actividad a realizar, esta fase es supremamente importante ya que en términos ingenieriles, puede ser nuestro "Callback", se iniciaría el desarrollo tan pronto termine esta fase, y a su vez sigue siendo recursivo en cada iteración, debido a que dependiendo del tiempo en semanas que se empleó para llevar a cabo la primer fase del proyecto, incluyendo con el prototipo funcional, sirve de medición para calcular el tiempo en semanas que llevara hacer los cambios solicitados por el cliente, de esa forma el cronometro está en marcha, corriendo en contra, lo que supone que se deba trabajar las 40 horas a la semana, siendo respetuosos con el calendario, con la velocidad

del proyecto que es demarcada por la velocidad del equipo de desarrollo y la cantidad de tareas pendientes en esta fase, por lo regular se debe intentar distribuir bien la carga en todos los días de la semana para evitar desiertos o colapsos.

FASE DE ITERACIONES

Esta fase abarca la serie de iteraciones que deben hacerse antes de entregar el resultado, resolviendo la mayor cantidad de errores que se pueda en cada iteración, lo cual no nos exime que por cuestiones de tiempo o de sobre esfuerzo, se entregue un prototipo funcional con algún tipo de error, este a su vez debe ser tenido en cuenta para trabajar en “Background” o segundo plano haciendo iteraciones individuales para corregir este y los demás errores que se pudieran identificar, sin permitir que dichos errores afecten el comportamiento del prototipo funcional al restarle calidad, siendo un proceso muy transparente de mejora continua; iteraciones de no más de tres semanas para resolver los problemas nuevos como los identificados sin resolver, calculando así con la velocidad del proyecto y la cantidad en semanas que llevara realizar las tareas específicas de las historias de usuario.

FASE DE PRODUCCION

Pruebas adicionales de funcionabilidad y respuesta, revisiones de rendimiento antes de la migración al entorno del cliente, inclusión de nuevas características dependiendo de las necesidades surgidas por la organización o cambios de última hora, en este punto las entregas se harán cada semana esperando pulir el producto final lo más que se pueda, en caso de funcionalidades extra no documentadas por el cliente a su debido tiempo serán solucionadas en la fase de mantenimiento generando un valor agregado al producto.

FASE DE MANTENIMIENTO

Mientras la primera versión se encuentra en producción, el sistema debe seguir iterando en la mejora continua, o en la adaptación de nuevas funcionalidades requeridas por el cliente, de esta forma la velocidad de desarrollo disminuirá debido a que se debe dividir el equipo en tanto a soporte como en la programación de las nuevas funciones o la optimización de las existentes.

FASE DE MUERTE

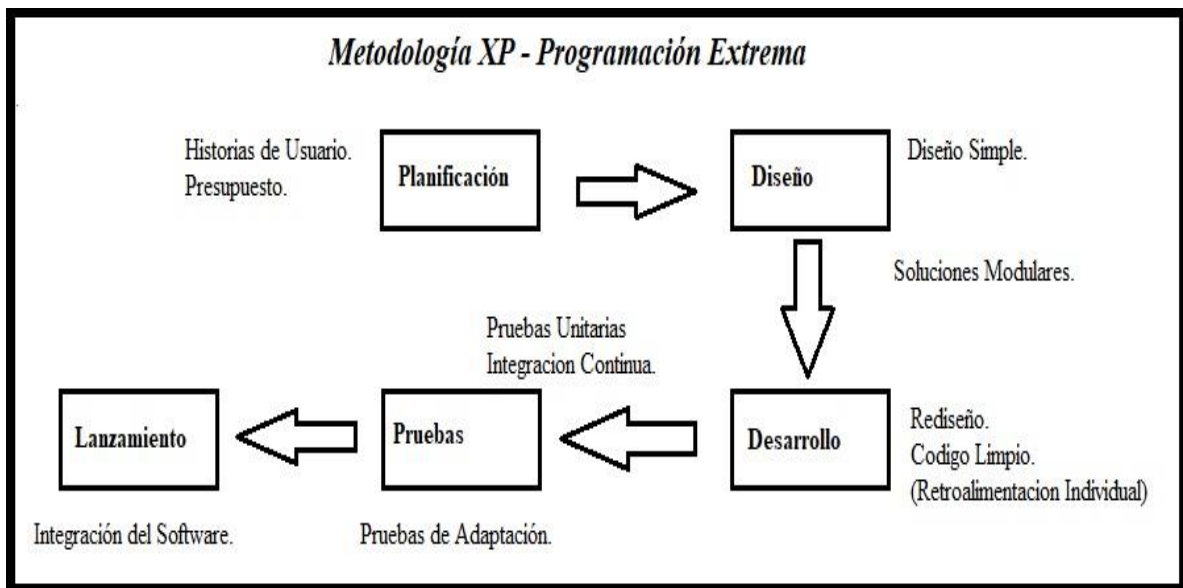
En este punto, el cliente no tendrá más historias para incluir en el sistema, culminando a cabalidad con los requerimientos especificados, esto resulta en que se satisfagan las necesidades del usuario. En otros aspectos como rendimiento y confiabilidad del sistema, abarcar tanto las historias funcionales como las no funcionales, brindando la seguridad, respaldo y estabilidad necesaria de un desarrollo software. Se generará entonces la documentación final del sistema y no se realizan más cambios en la arquitectura siendo este el reléase de esta aplicación.

ENCAPSULAMIENTO EN FASES FUNCIONALES.

Es de aclarar que en este punto del documento y para facilidad de la abstracción de los diferentes elementos que interactúan en esta metodología, simplificaremos un poco las cosas al nivel de tomar solo 4 fases totalmente funcionales, estas son similares a las etapas de la metodología Rational United Proccess (**RUP**) (IBM, 2001) o cualquiera otra metodología robusta que se base en iteraciones, sin importar su tiempo de retorno (véase Figura 6).

Figura 6.

Descripción grafica de la Metodología Extreme Programming. (XP)



Fuente: Elaboración propia.

METODOLOGÍA DEL PROYECTO

Teniendo en cuenta lo definido anteriormente en cuanto al encapsulamiento en fases funcionales para el proceso de desarrollo de la aplicación web, se requiere realizar una planeación de manera explícita sobre los procesos específicos a enmarcar en este punto, se deben levantar las respectivas historias de usuario para calcular mediante la necesidad planteada una estimación del tiempo pertinente para realizar a cabalidad dicha tarea, la redacción de las primeras historias de usuario es de vital importancia para el desarrollo del proyecto ya que determinara el enfoque del desarrollo y las prioridades del cliente. Siguiendo esta premisa de desarrollo (XP) debemos definir las necesidades prioritarias tanto del cliente como del desarrollo mismo, tenemos entonces:

Fase de Planeación

Al iniciar el proyecto con los requerimientos o solicitudes plasmados en la(s) historia(s) de usuario se realiza el presupuesto tanto monetario como de trabajo, en este caso horas trabajadas para cumplir una meta u objetivo. por su carácter modular esta metodología nos permite establecer las herramientas necesarias para llevar a cabo la totalidad del desarrollo, ahora bien, el algoritmo *BackPropagation* en su medida de eficiencia compara la salida arrojada por la **RNA** con los datos de entrenamiento que debiere “aprender” en la medida que itera buscando minimizar el valor del error y a su vez optimizar los valores de entrada de dicha función.

para esta investigación, la medida de eficiencia será impuesta por la correcta resolución de problemas documentados en bibliografías de soporte, sin embargo no se generaran datos de entrenamiento para la red neuronal, ya que se implementaran métodos heurísticos que mediante la conjugación de técnicas como el *gradiente descendente*, el algoritmo *BackPropagation*, su variación *Resilient BackPropagation (RProp)*, y los conceptos de *la teoría del error* logran que sin necesidad de una salida esperada para comparar y ajustar los pesos sinápticos de las neuronas, estas recorran puntos aleatorios de la función objetivo propagando el error hacia atrás, ajustando mediante la derivada la dirección y con RProp la magnitud del ajuste desde diferentes puntos en busca de la maximización o minimización de la función objetivo.

Nota: En la definición formal de la funcionalidad de este proyecto se especifica que para efectos académicos se resolverán problemas de optimización no lineal con máximo 5 incógnitas y 3 restricciones, cabe resaltar que la solución de PNL para las empresas de la región están sujetas a generar una función objetivo que enmarque y modele sus procesos organizacionales para luego hacer uso de la aplicación web y optimizar dicha función. En la documentación de la investigación se encontraron aplicaciones tanto en el sector de la construcción, como en el sector agrícola, ambiental y empresarial.

De igual forma se asignan los roles pertenecientes a la metodología (XP) y se establecen las historias de usuario iniciales a trabajar en esta fase, debido a que el proyecto no establece un cliente, los requerimientos funcionales y no funcionales se levantan en base de la problemática a resolver, haciendo del objetivo general su *Historia de Usuario Principal*, es de aclarar que este no será el único requerimiento a dar solución, más bien este será, el requerimiento global de la aplicación web, en él se enmarcaran las diferentes funcionalidades a desarrollar individualmente y por consiguiente estas funciones deben ir enmarcadas en *Historias de Usuario Individuales*, tenemos entonces:

Tabla 4.

Asignación de roles XP del proyecto.

Miembro	Roles XP	Metodología
W. German Aguirre B.	Manager	XP
	Desarrollador	
	Tester	

Fuente: Elaboración Propia

Levantamiento de Historias de Usuario

Tabla 5.

Historia de Usuario Principal

Historia de Usuario	
Numero: 1	Usuario: Objetivo General
Nombre Historia: Desarrollo de un aplicación web que mediante el uso de Redes Neuronales Artificiales resuelva Problemas de Optimización no Lineal Restricta	
Prioridad en el Proyecto: Alta	Riesgo en desarrollo: Vital
Puntos estimados: N/A	Iteración asignada: N/A
Programador Responsable: W. German Aguirre B.	
Descripción: La funcionalidad global de esta aplicación web se enmarca en llevar a feliz término la consecución de los objetivos del proyecto, por tanto este requerimiento es de importancia vital pues encapsula dentro de él las demás funcionalidades e historias de usuario correspondientes que se generaran a lo largo de su desarrollo	
Observaciones: No se le asignan puntos ni una iteración definida pues al ser el requerimiento global este debe verse reflejado al culminar el desarrollo.	

Fuente: Elaboración propia

Fase de Diseño

Para empezar con el diseño de la aplicación es de vital importancia recordar que, debido a lo ágil de este desarrollo, y de los conceptos inherentes de esta metodología, el diseño inicial se verá afectado en las constantes iteraciones que se pudieran presentar, por tanto, se deben elegir diseños sencillos e implementaciones básicas para facilitar su posterior adaptación a los cambios.

Tabla 6.*Diseño de la Base de Datos*

Historia de Usuario	
Numero: 2	Usuario: Desarrollador
Nombre Historia: Diseño de la Base de Datos	
Prioridad en el Proyecto: Alta	Riesgo en desarrollo: Vital
Puntos estimados: 5	Iteración asignada: 1
Programador Responsable: W. German Aguirre B.	
Descripción: Diseñar la Base de Datos de la Aplicación teniendo en cuenta que se almacenaran datos de usuario, sesión, datasets de prueba.	
Observaciones: En la primera iteración se espera una base de datos sencilla, esta también se adaptara con las iteraciones.	

Fuente: Elaboración Propia

Fase de Desarrollo

En este punto del proyecto cabe resaltar que en base a la *Historia de Usuario número 1*, se generaron diferentes requerimientos para llevar a cabo el objetivo primordial de este desarrollo, como lo son:

- Gestión de Usuarios.
- Gestión de Roles.
- Gestión de Tareas
- Gestor de soluciones
- Analizador de Ecuaciones
- Gestor de aprendizaje neural

Siendo estos partes fundamentales del desarrollo de la aplicación web en este modelo ágil e iterativo, generando una plataforma cada vez más robusta, modular y funcional que permitirá en la medida de su desarrollo lograr dar solución a la pregunta problema de esta investigación, además de estos se generaron otras historias de usuario cuya prioridad no es alta como lo son:

- Desarrollo de módulos independientes de PNL
- Solución de PNL con restricciones
- Despliegue del proyecto. (Beta)

Tabla 7

Gestión de Usuarios

Historia de Usuario	
Numero: 3	Usuario: Desarrollador
Nombre Historia: Gestión de Usuarios	
Prioridad en el Proyecto: Alta	Riesgo en desarrollo: Vital
Puntos estimados: 7	Iteración asignada: 1
Programador Responsable: W. German Aguirre B.	
Descripción: Desarrollar una plataforma web sencilla y escalable que permita el registro, modificación, visualización y eliminación de usuarios del sistema, validando su acceso mediante sesiones, brindando la opción de ingresar y registrarse	
Observaciones: En la primera iteración se espera una aplicación web sencilla, esta también se adaptara con las iteraciones.	

Fuente: Elaboración Propia

Fase de Pruebas

Para garantizar la usabilidad de la aplicación en diferentes entornos se realizarán pruebas en cada iteración con el fin de minimizar la presencia de errores en los módulos o formularios que llegaren a provocar un colapso de la aplicación web, por tanto, se generó la siguiente historia de usuario

Tabla 8

Realización de Pruebas

Historia de Usuario	
Numero: 4	Usuario: Desarrollador
Nombre Historia: Realización de Pruebas	
Prioridad en el Proyecto: Alta	Riesgo en desarrollo: Medio
Puntos estimados: 6	Iteración asignada: N/A
Programador Responsable: W. German Aguirre B.	

Descripción: Realizar una serie de pruebas a la aplicación tanto a los campos de los formularios como a las validaciones que estas pueden necesitar, realizar iterativamente a medida que avanza el proyecto

Observaciones: Empezar en la Iteración 2 y ciclar conforme el proyecto.

Fuente: Elaboración Propia

Fase de Despliegue

Al realizar todas y cada una de las historias de usuario resultantes, (las explícitas y las implícitas) abordando también las pruebas iterativas que se le realizaron a la aplicación, notándose que respondía de manera positiva a las pruebas realizadas, y resolviendo de manera satisfactoria los problemas de optimización de las literaturas de referencia. Generando mediante las iteraciones una aplicación estable y robusta, se despegará la aplicación web en un servidor público para su implementación por los usuarios que la llegaren a necesitar siendo esta la última *Historia de Usuario* a resolver, es de aclarar que aun en despliegue la aplicación es susceptible a modificaciones y mejoras que se lleguen a requerir.

Tabla 8

Realización de Pruebas

Historia de Usuario	
Numero: 5	Usuario: Desarrollador
Nombre Historia: Despliegue de Aplicación WEB	
Prioridad en el Proyecto: Alta	Riesgo en desarrollo: Medio
Puntos estimados: 10	Iteración asignada: N/A
Programador Responsable: W. German Aguirre B.	
Descripción: Realizar el despliegue de la aplicación cuyo código se almaceno previamente en https://www.gitub.com/GermanCode/Germath.js se desplegara en https://www.GermathJS.herokuapp.com con la cuenta de Estudiante de la Universidad de la Amazonia	
Observaciones: Despliegue Final	

Fuente: Elaboración Propia

CRONOGRAMA

OBJETIVOS	TIEMPO DE EJECUCION															
	MES 1				MES 2				MES 3				MES 4			
OBJETIVO ESPECIFICO 1	SEMANA															
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Determinar mediante la revisión bibliográfica el algoritmo de solución más apropiado para resolver problemas de Optimización No Lineal con Restricciones.	X															
Establecer el tipo de RNA necesaria para abarcar la totalidad del problema		X														
Identificar el método de aprendizaje óptimo a impartir sobre la red neuronal para solucionar problemas PNL.			X													

OBJETIVO ESPECIFICO 2	SEMANA															
	MES 1				MES 2				MES 3				MES 4			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Planificación:																
Historias de Usuario				X												
Revisión del estado de las Historias				X												
Confirmación de las Historias					X											
Diseño:																
Diseño de la Base de Datos.					X											
Revisión, ajustes a la base de datos.																
Desarrollo:																
Gestión de Usuarios.					X	X										
Gestión de Roles.						X	X									
Gestión de Tareas							X	X					X			
Gestor de soluciones								X					X	X		
Analizador de Ecuaciones								X	X					X	X	
Gestor de aprendizaje neural									X	X				X	X	
Desarrollo de módulos independientes de PNL										X	X	X		X	X	
Solución de PNL con restricciones												X				
Solución de PNL sin restricciones												X	X			
Despliegue del proyecto. (Beta)													X	X		
Pruebas:																
A medida que se desarrolla la Aplicación Web, se aplicaran una serie de pruebas funcionales e iterativas.															X	
Documentación:																
Manual de usuario.			X				X					X				

OBJETIVO 3	SEMANA															
	MES 1				MES 2				MES 3				MES 4			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Despliegue e implementación funcional de la aplicación mediante ejercicios de prueba de la documentación y en diferentes empresas de la región.																X

REFERENCIAS

- Allende, C. M., & Bouza, C. (05 de 2005). Professor George Bernard Dantzig, Life And Legend. *Revista Investigcion Operacional*, 205-211. Recuperado el 09 de 2019
- Anzola, N. S. (2015). Máquinas de soporte vectorial y redes neuronales artificiales en la predicción del movimiento usd/copspot intradiario. *ODEÓN*, 113-172.
- Arévalo, W., & Atehortúa, A. (2012). Metodología de Software MSF en pequeñas empresas. *ACTIVA*, ISSN 2027-8101.(4), 83-90.
- Beck, M. B.-K. (2001). *Manifiesto Ágil*. Obtenido de <https://agilemanifesto.org/iso/es/manifesto.html>
- Blazquez, J. R. (2005). El Perceptron. *Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada*, 16.
- Boeree, D. C. (2009). *La Neurona*. (Universidad de Shippensburg) Recuperado el 2019, de <http://webspace.ship.edu/cgboer/genesp/neuronas.html>
- Boirivant, J. A. (2009). La Programación Lineal Aplicación De La Pequeñas y Medianas Empresas. *Reflexiones*, 88, 89 - 105.
- Cadavid, A. N., Martínez, J. D., & Vélez, J. M. (2013). Revisión de metodologías ágiles para el desarrollo de software. *Prospect*, 11(2), 30 - 39.
- Calvo, D. (08 de 12 de 2018). <http://www.diegocalvo.es/perceptron/>. Recuperado el 09 de 2019, de <http://www.diegocalvo.es/perceptron/>
- Caparrini, F. S. (26 de 12 de 2018). *Redes Neuronales: una visión superficial*. (Dpto. de Ciencias de la Computación e Inteligencia Artificial) Recuperado el 09 de 2019, de <http://www.cs.us.es/~fsancho/?e=72>
- Daniel David Montenegro Murillo, M. A. (2018). Using Artificial Neural Networks to predict monthly precipitation for the Cali river basin, Colombia. *DYNA UNAL COLOMBIA*, 86(211), 122-130.
- DeepAI. (17 de 05 de 2019). *BackPropagation*. (DeepAI) Recuperado el 09 de 2019, de <https://deepai.org/machine-learning-glossary-and-terms/backpropagation>
- Gely, M. C. (2009). Métodos matemáticos de optimización no restringida Búsqueda multivariable. *Universidad Nacional del Centro de la Provincia de Buenos Aires*, 19.
- Gil, M. C. (2008). Aplicacion Web. *Departamento de Ingeniería Electrica, Electronica y de control*, 48-58. Recuperado el 04 de 2020
- Goic, M. (2005). Optimización de Problemas no lineales. *Universidad de Chile*, 15.
- Graupe, D. (2007). *Principles Of Artificial Neural Networks*. World Scientific.
- Gutierrez, C. S. (2019). Problemas de Asignación Generalizada: modelización, aplicaciones lógicas y métodos de solución. *Universidad de Valladolid*, 1(1), 92.
- Gutierrez, J. M., & Pulido, J. A. (2015). The resource constrained project scheduling Assuming multiobjective metaheuristics to solve a three-objective optimisation problem for relay Node deployment in Wireless Sensor Networks. *ELSEIVER*, V, 13.
- Hassim, Y. M., & Ghazali, R. (2012). Training a Functional Link Neural Network Using an Artificial Bee Colony for. *ReserchGate*, 7.

- Héctor, M., & Briceño, G. (2011). Modelo de Programación Lineal Aplicado a la Red Logística de una Empresa del Sector Plástico. *Corporación Universitaria Minuto De Dios*, 34.
- Henao, J. D., & Dumar, M. A. (2007). Modelado Del Precio Del Café Colombiano en la Bolsa de Nueva York Usando Redes Neuronales Artificiales. *Revista Facultad Nacional de Agronomía Medellín*, LX(2), 4129-4144.
- Hernández, C. M. (2017). Redes Neuronales para Clasificación: Una aplicación al caso de Riesgos Laborales en Colombia. *PONTIFICIA UNIVERSIDAD JAVERIANA*, 72.
- Hernandez, S. P. (2019). Optimización no lineal. En *OPTIMIZACIÓN Y CONTROL ÓPTIMO* (pág. 29). Cartagena, Colombia: Universidad Politecnica de Cartagena.
- Hidalgo, L. D., & Díaz, H. H. (2010). Aplicación de un modelo de programación lineal en la optimización de un sistema de planeación de requerimientos de materiales (MRP) de dos escalones con restricciones de capacidad. *Ingeniería e Investigación*, 30(1), 168 - 173.
- IBM. (01 de 12 de 2001). *Rational Unified Process: Best Practices for Software*. (IBM) Recuperado el 20 de 09 de 2019, de Link
- Ínkaya, T., Kayaligil, S., & Özdemirel, N. E. (2014). Ant Colony Optimization based Clustering Methodology. *ELSEIVER*, V, 51.
- Jaramillo, V. H., Vera, D. S., & Barcia, K. (2018). Modelo de Programación Lineal Aplicado a una Empresa PYME de Calzado. *LACCEI - International Multi-Conference for Engineering*, 10.
- Larranaga, P., Inza, I., & Moujahid, A. (2005). Tema 8. Redes Neuronales. *Departamento de Ciencias de la Computacion e Inteligencia Artificial*, I, 19.
- Letelier, P., & Penadés, M. C. (2012). Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). *Universidad Politécnica de Valencia.*, 17.
- Lin Li, Z. Y. (2014). Evacuation dynamic and exit optimization of a supermarket based on particle swarm optimization. *Physica A*, 23.
- Lipicia Munguía Ulloa, M. A. (2005). *Investigacion De Operaciones*. Costa Rica: Editorial Universidad Estatal a Distancia.
- Martínez, A., & Martínez, R. (2010). Guía a Rational Unified Process. *Universidad de Castilla la Mancha*, 15.
- Mejía, G., & Elkin, C. (2007). Optimización del proceso logístico en una empresa de colombiana de alimentos congelados y refrigerados. *Revista de Ingeniería - Universidad de los Andes*(26), 8.
- Menna, S. (2014). Heurísticas y Metodología de la Ciencia. *Mundo Siglo XXI*, IX(32), 67-77.
- Mibelli, G. P. (2005). Modelo Matematico de Programacion No-Lineal Para la Optimizacion de los Costos de Fabricacion de Vigas de Concreto Armado. *Escuela de Ingenieria Civil*, I, 72.
- Molina, S. G. (2013). Metodologías Ágiles Enfocadas Al Modelado de Requerimientos. *Universidad Nacional de la Patagonia Austral*(ISSN: 1852 - 4516), 16-17.

- Olabe, X. B. (2010). REDES NEURONALES ARTIFICIALES Y SUS APLICACIONES. *Escuela Superior de Ingeniería de Bilbao, EHU, II*, 79.
- Ospitia, D. F., & Cortes, L. K. (2019). Optimización de las Utilidades en la Empresa DM&E S.A.S mediante un Modelo de Programación Lineal que permita mejorar su Rendimiento Operacional. *Universidad Piloto de Colombia - Girardot*.
- Pitts, W., & McCulloch, W. S. (1943). A Logical Calculus of the Ideas Immanent in Nervous Activity. *Mathematical Biophysics From The University Of Illinois, College Of Medicine, V*, 19.
- Ploskas, N., & Samaras, N. (2014). Efficient GPU-based implementations of simplex type algorithms. *ELSEIVER, I*, 19.
- Riedmiller, M. (1994). Advanced supervised learning in multi-layer perceptrons — From backpropagation to adaptive learning algorithms. *Elseiver, XVII(3)*, 265 - 278.
- Sarmiento, L. H. (2018). Modelo de Optimización Multi-Objetivo para la Propagación de la Producción Agrícola a Pequeña Escala en Santander, Colombia. *Escuela de Estudios Industriales Y Empresariales*, 1117.
- Sepúlveda, G. F., Avilez, D. A., & Jaramillo, I. E. (2014). Análisis Del Precio Del Carbón Mediante Redes Neuronales Artificiales (RNA). *Boletín Ciencias de la Tierra(35)*, 31-36.
- Serna, M. D., Serna, C. A., & Ortega, G. P. (2010). Uso de la programación lineal paramétrica en la solución de un problema de planeación de requerimiento de materiales bajo condiciones de incertidumbre. *Ingeniería E Investigacion, 30(3)*, 96-105.
- Shiffman, D. (2012). *The Nature of Code: Simulating Natural Systems with Processing*. PapperBack.
- Staff, F. (2020). En 2020 Colombia tendría 50,3 Millones de Habitantes. *Forbes Colombia, 2*.
- Torres, L. C. (2011). *Redes Neuronales Artificiales*. Recuperado el 04 de 2020, de <https://disi.unal.edu.co/~lctorress/RedNeu/LiRna007.pdf>
- Trujillo, M. A. (2019). Aproximación A Un Modelo De Programación No Lineal Para La Asignación De Tráfico En La Ciudad De Pereira. *Universidad Libre de Pereira*, 44.
- Universidad de Murcia. (2014). Redes neuronales biológicas - [Imagen]. En A. Requena, R. Quintanilla, J. Bolarín, A. Vázquez, A. Bastida, J. Zúñiga, & L. Tomás., *Nuevas Tecnologías y Contaminación de Atmósferas, para PYMES* (págs. VI - 1). España: Universidad de Murcia. Recuperado el 09 de 2019, de <https://www.um.es/LEQ/Atmosferas/Ch-VI-3/C63s4p1.htm>
- Universidad Nacional Autónoma de México. (2002). Metodologías y procesos de análisis de software. En *Ingeniería de Software* (págs. 11-16). México.
- Valdes, M. M., Aleaga, A. M., & Vidal, G. G. (2014). Redes neuronales artificiales en la predicción de insolvencia. Un cambio de paradigma ante recetas tradicionales de prácticas empresariales. *Universidad Tecnológica Equinoccial, V(2)*, 21.

- Villavicencio, A., Arumí, J. L., & Holzapfel, E. (2011). Planificación de recursos hídricos en zonas de secano usando un modelo de optimización no lineal. *Obras y Proyectos*, 10, 73-80.
- Wang, Y. (2014). The Hybrid Genetic Algorithm with two Local Optimization Strategies for traveling salesman problem. *ELSEIVER*, LXX, 16.