UNIVERSIDAD NACIONAL DE SAN AGUSTÍN

ESCUELA DE POSGRADO

UNIDAD DE POSGRADO DE LA FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS

Nuevo Modelo de Red Neuronal para Aprendizaje Supervisado Basado en Aprendizaje por Refuerzo con Valores de Influencia

Tesis presentada por el bachiller:

André Mauricio Valdivia Ballesteros

Para optar el Grado Académico de Maestro en Ciencias **Informática**, mención en **Tecnologías de Información**

Asesor: **Dr. Jose Alfredo Herrera Quispe**

AREQUIPA - PERÚ

2018

AGRADECIMIENTOS

- A mi esposa, por la paciencia, apoyo y sacrificio que tuvo para que pueda terminar este trabajo de la mejor manera;
- A mi madre, que durante muchos años lucho arduamente para sacarme adelante, que se sacrificó mucho por que llegue hasta donde estoy hoy;
- A mi padre, por su constante preocupación, ayuda y motivación en que termine este trabajo;
- A mi asesor Dr. Jose Alfredo Herrera Quispe, por su confianza y exigencias para con este trabajo;
- A mi asesor externo Dr. Dennis Barrios Aranibar, por todo el conocimiento y dedicación impartida y estar siempre disponible en ayudar en los momentos más difíciles;
- Al profesor Mg. Alexander Ocsa Mamani, que dedicó tiempo en compartir sus conocimientos y así sentirme confiado de mis conocimientos en computación;
- A los profesores de la maestría, por su enseñanza y su grande motivación en seguir adelante durante todo el proceso de la maestría;
- A CONCYTEC y CIENCIACTIVA, por el apoyo y financiamiento de esta investigación;
- A todos mis familiares y amigos que directa o indirectamente contribuyeron en finalizar este trabajo de investigación.
- Finalmente darle la gloria a Dios, por todo lo que me da sin merecerlo.

RESUMEN

La auto-organización neuronal es una característica innata en los cerebros de los mamíferos, y es muy necesaria para su operación. Los modelos de redes neuronales artificiales más conocidos que usan esta característica son los mapas auto-organizados (SOM) y las redes de teor´ıa de resonancia adaptativa (ART), pero estos modelos, no toman a la neurona como una unidad de procesamiento, como su contraparte biológica; además que son modelos mayormente usados para el paradigma de aprendizaje no supervisado, esto quiere decir, que no se cuenta con modelos robustos auto-organizados en el paradigma de aprendizaje supervisado. En otro sentido, el paradigma de aprendizaje por refuerzo con valores de influencia, usado en sistemas multi-agentes, prueban que los agentes se pueden comunicar entre ellos, y que pueden auto-organizarse para asignarse tareas, sin ninguna interferencia.

Motivados por estas caracter´ısticas faltantes en las redes neuronales artificiales, y con el algoritmo de aprendizaje por refuerzo con valores de influencia se propone un nuevo modelo de red neuronal enfocado en el resolver problemas de aprendizaje supervisado, usando a los agentes de aprendizaje por refuerzo como neuronas en nuestro modelo; modelo que tiene como una caracter´ıstica importante las diferentes funciones de activación, dado que son únicas para cada neurona. Esta característica es importante también para la auto-organización.

Los agentes neuronales funcionarán en un espacio discreto, además de usar un algoritmo de aprendizaje distinto a la propagación del error, el cual es usado en muchas redes. Se propone un algoritmo inspirado en la forma en que las redes SOM propagan su conocimiento, y de esta forma los estados aledaños al estado entrenado puedan adquirir el conocimiento de este.

Para probar la funcionalidad de este modelo, se usaron bases de datos de baja dimensionalidad, y se comparó su desempeño con un perceptrón multicapa, donde en la mayoría de las bases de datos se mejoró el rendimiento de este. La creación de este nuevo modelo, es la base para posteriores investigaciones, donde la importancia fundamental de este trabajo es el nuevo concepto de neurona.

Palabras clave: Redes neuronales, Multiagentes, Aprendizaje por Refuerzo, Auto-organización.

ABSTRACT

Neural self-organization is an innate feature of the brains of mammals and it is very necessary for their operation. The most known artificial neural network models that use this characteristic are the Self-Organized Maps (SOM) and the Adaptive Resonance Theory (ART), but these models do not take the neuron as a processing unit, as it's biological counterpart does; besides these are models mostly used for the unsupervised learning paradigm, this means that there aren't robust self-organized models in the supervised learning paradigm. In other way, influence value reinforcement learning paradigm, used in multi-agent systems, prove that agents can communicate among them, and can self-organize themselves to assign tasks, without interference.

Motivated by the lack of features in the artificial neural networks, and taking into account the influence values reinforcement algorithm, a new neural network model is proposed, which is focused on solving supervised learning problems by using reinforcement learning agents as neurons in our model; model that has the different activation functions as an important characteristic, because these are unique for each neuron. This is also an important feature for self-organization.

The neural agents will work in a discrete space, besides using a learning algorithm different from the backpropagation, which is used in many networks. An algorithm inspired in the way the SOM networks propagate their knowledge is proposed, this way the neighboring states to the trained state can acquire its knowledge.

In order to prove the functionality of this model, low dimensionality daya bases were used and their performance was compared by a multilayer perceptron, where in most of the databases its performance was improved. The creation of this new model is the base for further importance of this investigation is the concept of neuron.

To prove the model functionality, we used databases of low dimensionality, and we compare its performance with multilayer perceptron, where in most of the databases the performance was improved. The creation of this novel model, is the base for further research, where the fundamental importance of this work is the novel neural concept

Keywords: Neural Networks, Multi-agents, Reinforcement Learning, Self - organization.

ÍNDICE GENERAL

1 INTRODUCCIÓN 1

1.1 Objetivos de la Investigación 3

1.1.1 General 3

1.1.2 Específicos 3

1.2 Descripción del problema 3

1.3 Relevancia del problema 4

1.4 Área de Investigación 4

1.5 Método propuesto 4

1.6 Organización del trabajo 5

2 REDES NEURONALES ARTIFICIALES 7

2.1 Conceptos básicos 7

2.2 Aproximación de funciones 8

2.2.1 Regresión Linear 9

2.2.2 Perceptrón 10

2.2.3 Perceptrón multicapa 11

2.2.4 Redes neuronales ART 12

2.3 Consideraciones Finales 14

3 APRENDIZAJE POR REFUERZO 15

3.1 Principales Componentes 15

3.2 Proceso de Decisión Markoviano 17

3.2.1 Retorno 18

3.2.2 Política 18

3.3 Funciones de valor 19

3.3.1 Aprendiendo las funciones de valor 20

3.3.2 Estrategias de selección de acciones 21

3.4 Sistemas multi-agente 22

3.4.1 Teoría de juegos 22

3.4.2 Aprendizaje por Refuerzo con Valores de Influencia 23

4 Trabajos Relacionados 27
4.1 Redes neuronales ... 28
4.2 Análisis desde la inspiración biológica ... 29
4.2.1 Red Neuronal Biológica ... 30
4.2.2 Auto-organización ... 33
4.3 Análisis desde su capacidad computacional ... 34
4.4 Aprendizaje por Refuerzo ... 38
4.4.1 Aproximación de funciones ... 38
4.5 Consideraciones Finales ... 39

5 Propuesta de Investigación 41
5.1 Modelo propuesto ... 41
5.1.1 Neurona ... 41
5.1.2 Discretización de los estados ... 42
5.1.3 Red Neuronal ... 42
5.2 Segunda Versión ... 44
5.2.1 Propagación del conocimiento ... 44
5.3 Intuiciones ... 45
5.4 Métricas de Evaluación ... 46

6 Resultados experimentales 51
6.1 Aspectos Preliminares ... 51
6.1.1 Bases de datos ... 52
6.1.2 Detalles de implementación ... 52
6.2 Validación del método propuesto ... 54
6.2.1 Jain ... 55
6.2.2 Flame ... 56
6.2.3 Pathbased ... 57
6.2.4 Spiral ... 59
6.2.5 Compound ... 61
6.2.6 Aggregation ... 63
6.2.7 R15 ... 65
6.2.8 Iris ... 68
6.2.9 Blood ... 69
6.2.10 Discusión ... 70
6.3 Análisis de parámetros ... 71
6.3.1 Alpha (α) ... 71

6.3.2 Beta (β) ... 72
6.3.3 Factor de Temperatura ... 73

6.3.4 Número de divisiones Propagación de estados 73
6.3.5 Estructura 75

7 CONCLUSIONES Y TRABAJOS FUTUROS 77
7.1 Conclusiones 77
7.2 Limitaciones 78
7.3 Trabajos futuros 78
7.4 Principales Contribuciones 78

1 INTRODUCCIÓN

Las redes neuronales artificiales son modelos computacionales inspirados en el sistema nervioso de los seres vivos. Estos tienen la habilidad de adquirir y mantener información y pueden ser definidos como un grupo de unidades de procesamiento, representadas por neuronas artificiales, interconectadas por muchas conexiones (sinapsis artificiales), implementadas por vectores y matrices de pesos sinápticos. A lo largo de los años se han creado diferentes modelos de redes neuronales, donde los principales motivos por los que se crearon estos nuevos modelos son tres: El aumento de la eficiencia en problemas ya resueltos, resolución de nuevos tipos de problemas y el asemejarse a la red neuronal biológica. Estos objetivos se han cumplido gracias a los diferentes modelos ya creados como las redes convolucionales o las redes de impulsos, sin embargo, el tercer objetivo el cual fue el motivo por el que nacieron las redes neuronales artificiales, tiene un amplio campo de investigación dado que existen características conocidas de las redes biológicas que no han sido implementadas artificialmente, y además de eso, aún nos falta mucho por conocer sobre el funcionamiento del cerebro humano.

A pesar de esto, una caracter´ıstica conocida y muy importante para el correcto funcionamiento de el cerebro es su capacidad de auto-organización. Esta cualidad se fundamenta en que no existe la suficiente información genética como para que la organización neuronal sea predefinida, sino que las neuronas se auto-organizan para que el cerebro pueda asociar neuronas y de esta manera funcionar de manera óptima. Inspirados en esta cualidad, se crearon distintos modelos como los mapas auto-organizativos o *self-organized map* (SOM) o como las del gas creciente neural o *growing neural gas*(GNG); estas redes toman a la neuronas como representaciones óptimas de un conjunto de datos, formando clústers entre ellas, sin embargo, las neuronas cerebrales no se comportan de esta manera, sino que son unidades de procesamiento que gracias a los neurotransmisores, se comunican con sus neuronas vecinas a través de impulsos formando sinápsis neuronales, y de esta manera auto-organizándose y especializándose para alcanzar objetivos grupales, los cuales

dependen del grupo de neuronas o región cerebral a la que pertenecen.

Por otro lado, el paradigma de aprendizaje por refuerzo se caracteriza por aprender con la interacción directa con el ambiente, en donde un agente se le premia o castiga de acuerdo a la acción tomada en un estado dado, es decir, causa-efecto, y donde el principal objetivo es maximizar la recompensa obtenida. Este paradigma, aumenta sus capacidades de resolver diferentes problemas gracias a la interacción de varios agentes en el sistema, estos se denominan sistemas multiagente. En el campo multiagente, se trata con un entorno aún más incierto, porque además del ambiente, tenemos que las acciones de otros agentes pueden alterar a su vez el entorno de los otros agentes, sin embargo este enfoque es aún más natural al mundo que conocemos.

Existen muchos enfoques para resolver problemas planteados para los sistemas multiagente. Uno de ellos es el paradigma de aprendizaje con valores de influencia, el cual se basa en la idea de que cada agente puede cambiar su comportamiento en función de la opinión de otros. Entonces, ya que esta propuesta está desarrollada en el contexto del aprendizaje a través de recompensas, el valor de las soluciones propuestas será dado por la recompensa individual obtenida por cada agente, y por la opinión que los otros agentes tienen sobre la solución que se dio individualmente. Esta opinión deberá ser una función de la recompensa obtenida por los agentes. Esto es, si el agente perjudica a otros agentes al hacer que ellos obtengan una recompensa menor a las recibidas anteriormente, ellos tendrán una opinión negativa de sus acciones . Además se demostró que este paradigma de aprendizaje permite la auto-organización de los agentes en sistemas complejos [2], propiedad por la cual los agentes pueden resolver el problema de asignación de tareas de forma óptima sin intervención externa en su organización.

Motivados por la característica de auto-organización neuronal, faltante en las redes neuronales artificiales, se modela a los agentes como neuronas con el objetivo de crear redes neuronales, donde cada neurona aprende como un agente de aprendizaje por refuerzo, e influencia en el aprendizaje de sus neuronas vecinas, aprendiendo así tanto de la respuesta del sistema, así como de la opinión de sus congéneres. Es importante recalcar que al modelar a la red de esta manera, las neuronas no solo serán neuronas con capacidades auto-organizativas y ser a su vez

unidades de procesamiento, sino que cada una tendrá un conocimiento diferente, caracter´ıstica que carecen los modelos tradicionales de redes neuronales artificiales

1.1 Objetivos de la Investigación

1.1.1 General

Crear un nuevo modelo de red neuronal discreto en donde las neuronas sean modeladas como agentes, por lo tanto la red neuronal sea un sistema multiagente, teniendo como principal característica la auto-organización de neuronas.

1.1.2 Específicos

- Estudio del estado del arte en modelos neuronales y aprendizaje por refuerzo en sistemas multiagente.
- Definir un nuevo modelo de neurona.
- Implementar un modelo neuronal discreto basado en el nuevo modelo de neurona.
- Validar y evaluar el modelo propuesto a través de las pruebas en problemas tradicionales y problemas creados espec´ıficamente para evaluar el desempeño de la red.

1.2 Descripción del problema

Las redes neuronales con aprendizaje supervisado permiten interpolar funciones, sin embargo, los modelos más conocidos como perceptrón de múltiples capas con su algoritmo backpropagation o redes de funciones de base radial (RBF) adolecen de auto-organización [21], es decir, la capacidad de las neuronas de asociarse para resolver distintos problemas; además, generalmente estas redes funcionan muy bien al interpolar una única función a la vez, lo cual les genera limitantes al desarrollar aplicaciones reales, como por ejemplo, al usarlas dentro de agentes, estos no son capaces en desenvolverse en múltiples tareas.

Por otro lado, el aprendizaje por refuerzo en sistemas multi-agente, permite que sus grupos de individuos, se puedan especializar en diferentes tareas, y además

permite soluciones bastantes plásticas en lo que refiere a lograr objetivos globales para todo el sistema. Lo anterior se ve especialmente resaltado cuando se aplican algoritmos de aprendizaje por refuerzo con valores de influencia [2].

En este sentido, en este proyecto proponemos crear una red neuronal modelada como si fuese un sistema de multiagente donde cada agente represente una neurona, las relaciones entre los agentes representen relaciones sinápticas entre neuronas y el proceso de aprendizaje de la red neuronal se maneje en base a un algoritmo de aprendizaje por refuerzo para sistemas multiagentes, en este caso específico se plantea utilizar el paradigma de aprendizaje por refuerzo con valores de influencia. Es así que este nuevo modelo neuronal posee la capacidad de interpolar funciones, así también como capacidad de auto-organización neuronal, propiedad que es innata en los algoritmos de aprendizaje por refurezo con valores de influencia.

1.3 Relevancia del problema

Una de las características fundamentales de las neuronas cerebrales es su capacidad de auto-organización, lo que significa que no existen instrucciones genéticas para la distribución, especialización ni conexión de estas. Por muchos años se desarrollaron nuevas redes neuronales que tienen una estructura estática, pero esto no se asemeja a la inspiración biológica del cerebro. Además en cuanto a aplicaciones, se necesita que las redes puedan simular en parte al cerebro humano, donde no solo se especializa en una sola tarea, sino en muchas otras. ¿Se podrá crear una red neuronal lo suficientemente plástica para interpolar diferentes funciones al mismo tiempo y no se necesite reentrenar la red al cambiar dichas funciones?

1.4 Área de Investigación

Machine learning es el área de investigación, donde el campo de investigación es redes neuronales y aprendizaje por refuerzo.

1.5 Método propuesto

Método en espiral, se empezará a implementar modelos simples hasta obtener un modelo estable.

1.6 Organización del trabajo

El presente trabajo está organizado de la siguiente manera:

- En el **Cap´ıtulo 2** se presenta el problema de aprendizaje supervisado formalmente, así como técnicas para el correcto modelaje de sistemas de aprendizaje. Además los modelos que más se asemejan al modelo propuesto.
- En el **Cap´ıtulo 3** se aborda el paradigma de aprendizaje por refuerzo. y la importancia de los sistemas multiagentes.
- En el **Capítulo 4** se hace una revisión del estado del arte de modelos de redes neuronales, su inspiración biológica y sus aplicaciones en el área computacional.
- En el **Cap´ıtulo 5** introduciremos el nuevo modelo propuesto de este trabajo.
- En el **Capítulo 6** se analiza el modelo con métricas para aprendizaje supervisado, así como una comparación del desempeño contra un perceptrón multicapa, y un análisis de sus parámetros más importantes.
- En el **Cap´ıtulo 7** se realiza las conclusiones del trabajo de tesis presentado, limitaciones, trabajos futuros, y principales contribuciones.

2 REDES NEURONALES ARTIFICIALES

En esta sección, se define muchos conceptos básicos relacionados con el aprendizaje máquina que son necesarios para entender este trabajo, además de otros modelos importantes auto-organizativos. Para poder ampliar estos conceptos aún más, se recomienda este compendio [46].

2.1 CONCEPTOS BÁSICOS

Como una introducción al aprendizaje supervisado o *supervised learning (SL)*, se presentan los siguientes conceptos básicos, sin embargo, se ampliarán los conceptos más adelante.

El aprendizaje supervisado o predictivo, tiene como principal objetivo el aprender a mapear entradas $\boldsymbol{x}$ a salidas y, dado un conjunto etiquetado de pares $D = (\boldsymbol{x}_i, y_i)_{i=1}^{N}$ donde D es llamado el *conjunto de entrenamiento*, y N es el numero de ejemplos de entrenamiento.

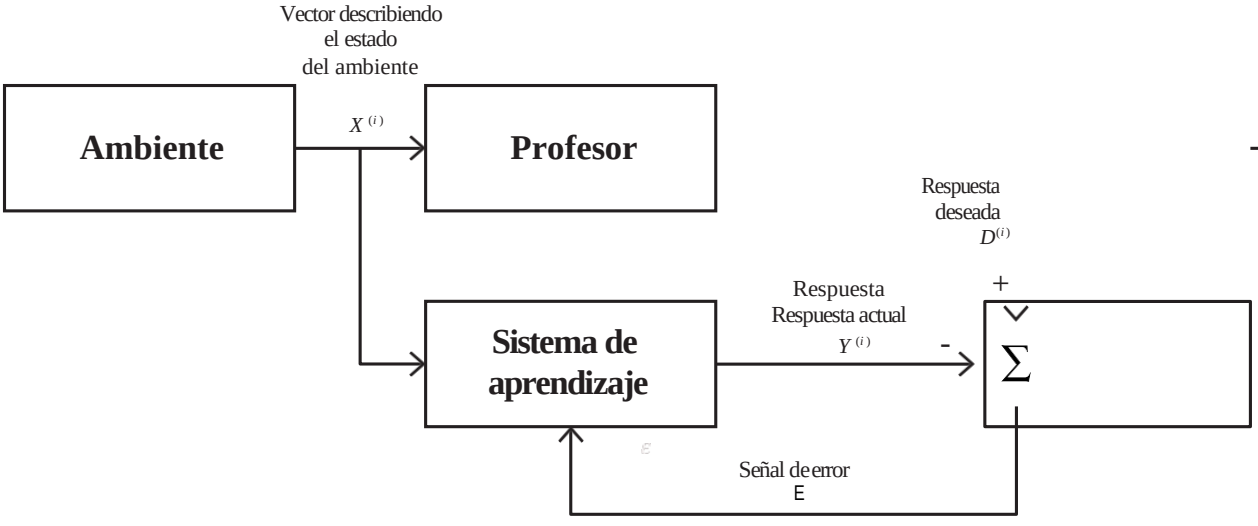


Figura 2.1: Aprendizaje Supervisado [25]

La figura 2.1 muestra su comportamiento más básico, donde cada entrada de entrenamiento $\boldsymbol{x}_i$ es un vector D-dimensional de números, también llamados atributos o caracter´ısticas, en general, $\boldsymbol{x} = [x_1, x_2, ..., x_d] \in \mathsf{R}$ puede ser un objeto estructurado complejo como una imagen, texto, mensaje, serie de tiempo, etc. Este vector de caracter´ısticas es expuesto tanto al profesor como al sistema de aprendizaje.

La variable de salida, en principio puede ser cualquier cosa, pero en la mayor´ıa de métodos, se asume que y_i es una variable *categórica* o *nominal* de un conjunto finito, $y_i \in [1, ..., C]$, o que es un valor real $y_i \in \mathsf{R}$. Cuando y_i es una variable categórica se dice que el problema es de *clasificación*, mientras que cuando es un valor real pertenece a un problema de *regresión*. Esta variable de salida sigue la función $f : X \longrightarrow Y$ para cada ejemplo de D, donde X es el espacio de entrada y Y es el espacio de salida. El profesor conoce esta función por lo que produce una respuesta deseada y_i, mientras que el sistema de aprendizaje debe de generar una función $\hat{f}(x) = \hat{y}$, que se asemeje lo más posible a $f(x)$, usando la señal de error E para esto. La señal de error en la mayoría de las redes, se usa el error cuadrático mínimo o *mean squared error* (MSE), dado por la ecuación 2.1.

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y - y)^2 \tag{2.1}$$

Existen muchas variantes de problemas dentro de SL, como el aprendizaje semi-supervisado, donde algunos ejemplos contienen una etiquete, muestras que otros no; o donde el vector x está incompleto. Se han detectado muchos problemas en este sentido, y es uno de los motivos, por los que se tienen diferentes modelos neuronales.

2.2 Aproximación de funciones

Cuando el espacio de entrada es pequeño $X \in [1, 2, 3]$, es fácil simplemente almacenar las posibles respuestas; desafortunadamente esto raramente sucede en la práctica.

La cantidad de funciones que pueden existir con entradas discretas, es mucha; peor es en el caso cuando trabajamos en un espacio continuo donde el tamaño de

funciones se vuelven infinitamente incontables. Como una solución a este problema, es usar la *aproximación de funciones*.

En lugar de explícitamente mapear cada posible entrada x a su correspondiente salida y, los aproximadores de funciones usan reglas y algoritmos para determinar y dado algún x. Es común hacer que los aproximadores de funciones dependan de *parámetros* $\theta = \{\theta_1, \theta_2, ..., \theta_n\}$, que son usados para calcular y dado x.

Notemos algunas cosas. Primero, aunque el número de funciones de la forma $f(x; \theta)$ no es estrictamente menor al número de funciones $f : \mathsf{R} \longrightarrow \mathsf{R}$, el número de diferentes configuraciones que una computadora tiene que intentar es efectiva- mente ahora, mucho más pequeño (para una cantidad de parámetros razonable).

Segundo, es muy probable que alguna distribución D no se ajuste del todo bien a una función $f(x; \theta)$.

Por ultimo, ahora tenemos una manera de explorar este subespacio de funciones de la forma $f(x; \theta)$, donde todo lo necesario es escoger un valor para cada θ_i y ver como se desempeña.

Usar aproximación de funciones, puede ser visto como una compensación entre encontrar exactamente funciones correctas, y el número de funciones que tenemos que intentar antes de encontrar la correcta. Esto es especialmente cierto, en el caso de espacios continuos.

2.2.1 Regresión Linear

Uno de los modelos más usados para regresión es conocido como *regresión linear*. En el caso de $f : \mathsf{R}^n \longrightarrow \mathsf{R}$, son típicamente denotados como:

$$f(x) = x^\mathsf{T} w$$

donde $x \in \mathsf{R}^n$ es el vector de entrada y $w \in \mathsf{R}^n$ es un vector de parámetros, o vector de pesos. Es común también añadir un termino de bias:

$$f(x) = x^\mathsf{T} w + \beta$$

donde $\beta \in \mathsf{R}$ es un parámetro escalar. Denotamos al conjunto de parámetros como $\theta = \{w, \beta\}$. En el caso de $f : \mathsf{R}^n \longrightarrow \mathsf{R}^m$ se tienen típicamente la siguiente notación:

$$f(x) = x^\mathsf{T} W + b,$$

donde $x \in \mathbb{R}^{n \times 1}$ es la entrada, $W \in \mathbb{R}^{n \times}$ es la matriz de pesos, y $b \in \mathbb{R}^{m}$ es el vector de bias.

2.2.2 Perceptrón

Es un modelo neuronal simple que toma un vector de entrada $\boldsymbol{x} = (x_1, x_2, ..., x_{n+1})$ y a través de un vector sináptico de pesos $\bar{w} = (w_1, w_2, ..., w_{n+1})$. La salida *out* es determinada por la ecuación 2.2, donde net es el producto punto entre los pesos y el vector de entrada, y f es la función de activación. Por convención, si hay n entradas al perceptrón, la entrada $(n + 1)$ se fija en -1 y el peso asociado a θ.

$$out = f(net) = f(\bar{w} \cdot \bar{x}) = f\left(\sum_{j=1}^{n+1} w_j x_j\right) = f\left(\sum_{j=1}^{n} w_j x_j - \theta\right) \tag{2.2}$$

En 1958 Rosenblatt introdujo un perceptrón discreto con una función de activación bipolar (Figura 2.2)

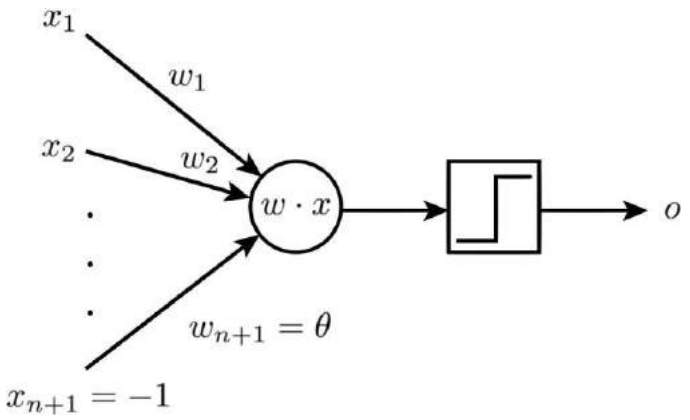


Figura 2.2: Ilustración esquemática de un modelo perceptrón

$$f(net) = sign(net) = \begin{cases} +1 & \text{if } net \geq \theta \iff \sum_{j=1}^{n} w_j x_j \geq \theta \\ -1 & \text{if } net < \theta \iff \sum_{j=1}^{n} w_j x_j < \theta \end{cases} \tag{2.3}$$

Los perceptrones pueden separar a los datos en dos clases, si las clases pueden ser separadas por un $n-1$ hiperplano. En otras palabras, el perceptrón puede resolver problemas separables linealmente como AND o OR. La función XOR, por otro lado, no es linealmente separable (Figura 2.3).

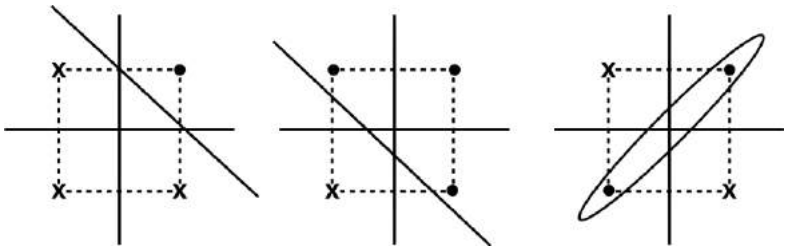


Figura 2.3: Problemas separables linealmente y no linealmente

2.2.3 PERCEPTRÓN MULTICAPA

Rumelhart [48] introdujo el método de la retropropapagación del error. El perceptrón multicapa es una generalización de la regla para muchas capas; no existe conexión entre neuronas de la misma capa, solo existe conexión entre neuronas de capas sucesivas (cada neurona de la capa l a cada neurona de la capa $l + 1$). La función de activación de activación más común es la siguiente:

$$f(net) = \frac{1}{1 + exp(-\gamma net)} \tag{2.4}$$

La constante $\gamma > 0$ determina la variación de la fuerza de la curva, y es usualmente seteada a 1. En el limite $\gamma \longrightarrow \infty$ tiende a la función de paso.

Considerando la red neuronal de una capa , los vectores de entrada y salida son $\bar{y} = (y_1, ..., y_j, ..., y_J)$ y $\bar{o} = (o_1, ..., o_k, ..., o_K)$, respectivamente, donde $o_k = f(net_k)$ y

$$net_k = \sum_{j=1}^{J} w_{kj} y_j \tag{2.5}$$

Seteamos $y_t = -1$ y $w_kJ = \theta_k$, un l´ımite para $k = 1, ..., K$ neuronas de salida. La salida deseada es $\bar{d} = (d_1, ..., d_k, .., d_K)$. Después de entrenar, se desea que to- das las muestras de la base de datos A_train, el modelo de salida muy cerca a los valores deseados (objetivo). El problema de entrenamiento es transformado en un problema de optimización, definiendo el error como:

$$E_p = \frac{1}{2} \sum_{k=1}^{K} (d_{pk} - o_{pk})^2 \tag{2.6}$$

donde p es el ´ındice de entrenamiento. E_p es la suma de los errores cuadrados de las neuronas de salida. Durante el aprendizaje, buscamos los pesos que minimicen E_p. Esto se hace usando el gradiente descendiente en E_p.

$$\Delta w_{kj} = -\alpha \frac{\partial E_p}{\partial w_{kj}} = -\alpha \frac{\partial E_p}{\partial (net_k)} \frac{\partial (net_k)}{\partial w_{kj}} = \alpha \delta_{ok} y_j \tag{2.7}$$

donde α es un learning rate positivo. Nótese que $-\partial E_p/\partial(net_k) = \delta_{ok}$, que es la formula generalizada de entrenamiento en la neurona k-th neurona de salida. La derivada parcial $\partial(net_k)/\partial w_{kj}$ es igual a y_j 2.5. Además,

$$\delta_{ok} = -\frac{\partial E_p}{\partial (net_k)} = -\frac{\partial E_p}{\partial o_k} \frac{\partial o_k}{\partial (net_k)} = (d_{pk} - o_{pk}) f'_k \tag{2.8}$$

donde f'_k denota la derivada de la función de activación con respecto a net_k y donde $f'_k = o_k(1 - o_k)$. La regla para actualizar el peso j-th de la neurona de salida k-th es dado por:

$$\Delta w_{kj} = \alpha (d_{pk} - o_{pk}) f'_k y_j \tag{2.9}$$

donde $d_{pk} - o_{pk} f'_k = \delta_{ok}$ es el error generalizado siguiendo atrás a través de las conexiones terminando en la neurona k-th.

2.2.4 Redes neuronales ART

Las redes neuronales sufren de un dilema que se llama : *Dilema de estabilidad - plasticidad*; que significa que un sistema debe de adaptarse a los cambios de entorno (debe ser plástico), pero el constante cambio puede hacer al sistema inestable, porque el sistema puede aprender nueva información solo olvidando lo que ya había aprendido. Este fenómeno, es un problema que las redes neuronales creadas en 1976, intentaron resolver.

Las redes ART (Adaptative Resonance Theory) incluyen una basta variedad de redes neuronales, tanto para aprendizaje supervisado como no supervisado. Sin embargo la forma básica de una ART sirve para aprendizaje no supervisado, a continuación se listarán las redes principales para aprendizaje supervisado y no supervisado, además de la imagen de referencia 2.4.

Las principales redes ART para aprendizaje no supervisado son:

- ART 1 [21], diseñado para patrones de entrada binarios,
- ART 2 [4], desarrollado para el clúster de patrones de entrada continuo,

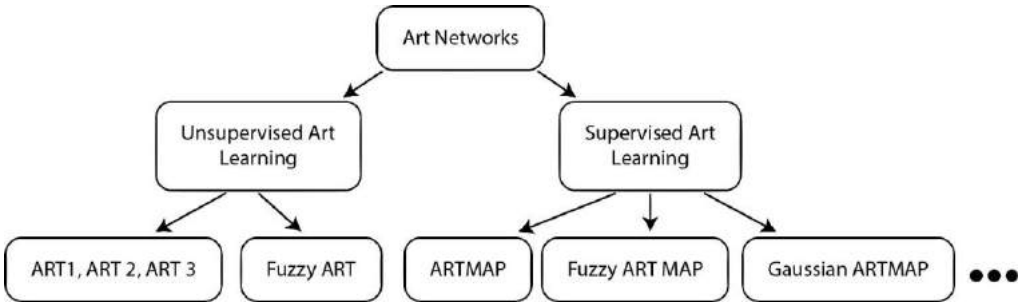


Figura 2.4: Taxonom´ıa de redes ART

- ART 3 [5], el refinamiento de estos modelos.
- Fuzzy Art [8], etc.

Las redes de aprendizaje supervisado ART, tienen como sufijo ”MAP”, donde la primera unidad toma la data de entrada y la segunda unidad toma la correcta data de salida; el algoritmo :

- ARTMAP [7],
- Fuzzy ARTMAP [6],
- Gaussian ARTMAP [57], etc.

Una ART en su modelo más simple, es un clasificador de vectores. Acepta como entrada un vector y lo clasifica en una categoría dependiendo en el patrón almacenado. Cuando el patrón es encontrado, es modificado para aceptar al nuevo vector. Si el vector de entrada no concuerda con ningún patrón con cierta tolerancia, entonces una nueva categoría es creada almacenando el nuevo patrón. Consecuentemente, ningún patrón almacenado es modificado a menos que concuerde con el vector de entrada con cierta tolerancia.

Las redes neuronales ART no son muy usadas hoy en d´ıa, dado que estas no poseen la propiedad estad´ıstica de consistencia, propiedad fundamental para poder generalizar la función que se quiere hallar con la red neuronal, dado que cuando el número de datos en la muestra tiende al infinito, la red no converge a su valor verdadero [50].

Se dice que un estimador o estadístico muestral es consistente cuando éste converge a su valor verdadero cuando el número de datos de la muestra tiende a infinito.

2.3 CONSIDERACIONES FINALES

En este cap´ıtulo se dieron dieron las bases para esta propuesta, definiendo el problema de aprendizaje supervisado, el cual es finalmente el que queremos atacar. Luego, se describió la aproximación de funciones como una solución para resolver este problema, dentro del cual se pueden encontrar muchos métodos como la regresión linear o el perceptrón, el cual es el principal componente desde donde se crearon las diferentes redes neuronales que existen hoy en día. Se describió el funcionamiento de el perceptrón multicapa, el cual es muy usado en la actualidad, y las redes ART, las cuales fueron creadas para resolver el dilema de estabilidad-plasticidad.

En el siguiente capítulo, ese hablará de otra rama de la inteligencia artificial llamado aprendizaje por refuerzo, el cual es muy necesario entender su funcionamiento dado que se usa como principio básico para nuestra propuesta de esta tesis.

3 Aprendizaje por Refuerzo

En esta sección se plantean las bases del paradigma de aprendizaje por refuerzo (RL), así como una base para entender los sistemas multi-agente (MAS). Se describirán las bases de los algoritmos usados en el modelo propuesto en esta tesis, necesarios para poder comprender de mejor manera la red neuronal propuesta.

- En la sección 3.1 presentaremos los dos componentes esenciales del aprendizaje por refuerzo.
- En la sección 3.2 se explicará el framework MDP, el cual es de suma importancia para el aprendizaje por refuerzo, además de entender conceptos como la recompensa acumulado y la política.
- En la sección 3.3 se explicará la importancia de las funciones de valor y valor-acción y las estrategias para poder aprenderlos.
- En la sección 3.4 se explicarán los sistemas con múltiples agentes, así como el algoritmo usado en la propuesta de tesis.

3.1 Principales Componentes

El aprendizaje por refuerzo o *Reinforcement Learning*, es un paradigma de aprendizaje dentro de la inteligencia artificial, que modela un problema de forma que pueda ajustarse a un escenario de agente-ambiente, siendo esto un enfoque distinto al como se modela un problema en aprendizaje supervisado. En SL, normalmente se tiene un modelo completo, o al menos uno con la información suficientemente necesaria para poder interpolar la función principal y poder predecir en base al sistema de aprendizaje o agente; mientras que en aprendizaje por refuerzo, el modelo es desconocido y cambiante dado que el agente, dependiendo de las acciones que este tome, el mundo donde este se encuentra cambia, lo que significa que debe de tomar acciones secuencialmente en el tiempo.

Los principales componentes dentro del paradigma de aprendizaje por refuerzo, son el **agente** y el **ambiente**. El agente como primer componente, formalmente, es una entidad independiente compuesta comúnmente por 3 subcomponentes:

- **Instrumentos Sensoriales:** Los agentes, necesitan percibir el mundo que los rodea, es decir, su entorno. Por ejemplo, si un agente se desenvuelve en fútbol de robots, necesita saber dónde esta la pelota, los arcos, entre otros objetivos u obstáculos, para poder tomar decisiones correctas. Mientras las tareas del agente se vuelvan más complejas, se requerirá de sensores aún más complejos. En el caso de un robot de rescate podrían requerirse sonares, cámaras, receptores de calor, etc.

- **Actuadores:** Cuando el agente pueda percibir su espacio, deberá de actuar, y cuando haga esto, influenciará en su entorno. En un simulador de fútbol, se necesita una función que posibilite moverse hacia una dirección dada. Es importante notar, que el hecho que influencien su entorno, es uno de los principales problemas cuando nos enfrentamos a problemas de múltiples agentes.

- **Núcleo lógico:** Por último, el agente necesita tomar las decisiones, de acuerdo al estado donde se encuentra. Para este objetivo, se implementan algoritmos de inteligencia artificial, donde el agente adquiere cierto grado de *inteligencia*, y así tener la capacidad de decidir correctamente, para poder alcanzar su objetivo.

El agente además necesita desenvolverse en un ambiente, también llamado entorno. El ambiente como segundo componente, delimita al agente, confinando las capacidades que este posee y dándole la información suficiente para que este pueda tomar las acciones necesarias para alcanzar el objetivo trazado.

La figura 3.1, nos muestra la información fluyendo en este paradigma, donde el agente reside en un ambiente. Este agente se encuentra en un estado actual s_t, en el cual, mientras no sea un *estado terminal*, debe de escoger una acción a_t. Al escogerla, el ambiente le proporcionará al agente el estado siguiente s_{t+1}, en el cual se encontrará por tomar la acción a_t. Además se le proporcionará una recompensa r_t, que dependerá si el agente está en buen camino o no. El principal objetivo del agente es aprender a maximizar la recompensa total.

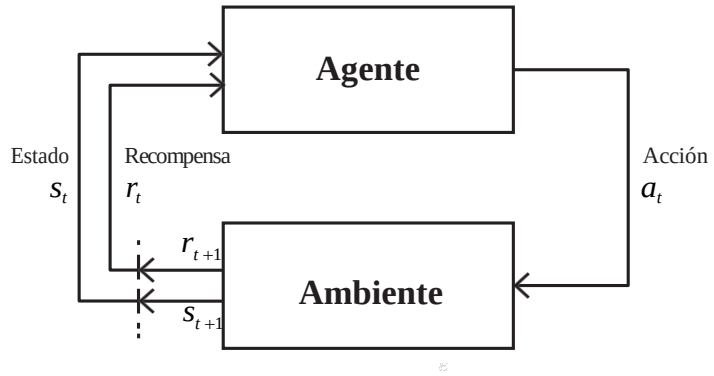


Figura 3.1: Aprendizaje por Refuerzo

3.2 Proceso de Decisión Markoviano

El Proceso de Decisión Markoviano o *Markov Decision Process (MDP)* [3], provee una interfaz matemática usada por los problemas en el aprendizaje por refuerzo, dado que ellos son adecuados para describir la información total de ambientes tanto estocásticos como determinísticos.

Para simplificar la explicación, se considerará el proceso estándar de aprendizaje por refuerzo. En su forma básica, un MDP, es matemáticamente definido en términos de una tupla (S, A, P, R), donde:

- S es el conjunto de todos los posibles estados que describen el contexto de ambiente, también llamado el *espacio de estado*;
- A es el conjunto finito de acciones que el agente puede tomar;
- $P: S \times A \times S \longrightarrow [0, 1]$ es una función de transición, un mapeo especificando la probabilidad $P^a_{s,s'}$ de llegar al estado s' cuando se toma la acción a en el estado s. Una suposición esencial hecha en MDP es que las dinámicas de la evolución de estado es *Markoviano*, es decir, la distribución de los siguientes estados es condicionalmente independiente del pasado, dado un estado actual.
- $R: S \times A \times S \longrightarrow \mathbb{R}$ es la función de recompensa. $R^a_{s,s'}$ describe una recompensa obtenida cuando el agente parte del estado s y llega al estado s' como resultado de tomar la acción a. La recompensa puede ser positiva o negativa, representando una utilidad o perdida respectivamente.

Los MDP, al ser usados en sistemas de aprendizaje por refuerzo, es necesario que cumplan con la propiedad de Markov, propiedad que se define de la siguiente manera:

$$P(s_t/s_{t-1}, a_{t-1}) = P(s_t/s_{t-1}, a_{t-1}, s_{t-2}, ..., s_0, a_0)$$

lo que significa que el estado siguiente solo depende del estado actual. Un agente que vive en un ambiente Markoviano solo necesita saber del estado más reciente, para poder tomar decisiones.

3.2.1 Retorno

En su forma más simple, el retorno es la suma de recompensas:

$$G_t = r_{t+1} + r_{t+2} + r_{t+3} + ... + r_T \tag{3.1}$$

donde T es un paso final de tiempo. Esto puede servir en ambientes donde exista un *estado terminal*. Pero en muchos casos, las tareas que tiene que hacer el agente no son *episódicas*, sino que no tiene un límite definido. Para estas tareas *continuas*, la ecuación 3.1 dado que $T = \infty$, y es por esto se necesita el concepto de descuen- to. Añadiendo este concepto, el nuevo objetivo del agente, debe ser maximizar el *retorno descontado* esperado, que está dado por :

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

donde γ es un parámetro , $0 \leq \gamma \leq 1$, llamado el factor de descuento.

3.2.2 Política

Si consideramos un agente sin memoria, el cual en cada momento de tiempo se le presente un estado y deba de escoger una acción determinada; este comportamiento se traduce en la política $\pi : \mathcal{S} \longrightarrow \mathcal{A}$. En este trabajo asumirá el uso de una política estocástica, la cual genera una distribución de probabilidad condicional, indicando la acción que tomaría el agente en un momento dado, la cual se define de la siguiente manera:

$$\pi(s, a) = P(a_t = a/s_t = s) \quad \text{y} \quad \sum_{a \in \mathcal{A}} \pi(s, a) = 1$$

Un concepto muy importante, es el de la *política óptima* π^*, la cual se define como la pol´ıtica que es mejor o igual que las otras pol´ıticas.

Sin embargo, en aplicaciones prácticas de aprendizaje por refuerzo, los estados y acciones son definidos por parámetros continuos como distancias o voltajes. Como resultado de esto, el conjunto $\mathcal{S}$ y $\mathcal{A}$ son t´ıpicamente largos o infinitos, y aprender la función de valor requiere de alguna forma de **aproximación de funciones**.

Existen dos variantes principales: MDPs con estados continuos y acciones discretas, y MDPs donde los estados y acciones son continuos. En esta tesis nos enfocaremos en el primero, usando el método de discretización el cual ampliaremos su concepto más adelante.

3.3 Funciones de valor

La gran mayor´ıa de algoritmos de aprendizaje por refuerzo involucran estimar el valor de la función, es decir, tener un concepto medible de cuan bueno para el agente es estar en un estado dado. Existen dos tipos de funciones: funciones de *valor* y funciones de valor-acción. El primero es definido como:

$$
\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi[G_t / S_t = s] \\
&= \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} / S_t = s \right] \\
&= \sum_a \pi(a/s) \sum_{s'} p(s'/s, a)[r(s, a, s') + \gamma v_\pi(s')]
\end{aligned}
$$

donde $\mathbb{E}_\pi$ denota le valor esperado dado que el agente sigue la pol´ıtica π, y t es cualquier momento en el tiempo. Es el valor esperado acumulativo, también conocido como retorno, que puede recibir en s y los siguientes estados del episodio que son amortizados con su respectiva tasa de descuento γ.

Mientras que v_π, mide cuan bueno es estar en un estado siguiendo la pol´ıtica π, q_π es la función de valor-acción de estar en un respectivo estado s y tomar la acción

a siguiendo la política π; la cual está definida de la siguiente manera:

$$
\begin{aligned}
q_\pi(s,a) &= \mathsf{E}_\pi[G_t / S_t = s, A_t = a] \\
&= \mathsf{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} / S_t = s, A_t = a \right] \\
&= \sum_{s'} p(s,s',a)\ r(s,a,s') + \gamma v_\pi(s')
\end{aligned}
\tag{3.2}
$$

Por otro lado, bajo la política óptima, las funciones de valor son llamadas *funciones de valor óptimas*, dado que estas representan que el agente escoge la acción óptima a cada paso de tiempo; y estas se denotan v_* y q_*:

$$v_*(s,a) = \max_\pi q_\pi(s,a) q_*(s,a) = \max_\pi q^\pi(s,a)$$

Si es que lo analizamos de mejor manera, podríamos escribir q_* en función de v_* de la siguiente manera:

$$q_*(s,a) = \mathsf{E}[r_{t+1} + \gamma v_*(s_{t+1}) / s_t = s, a_t = a]$$

En un MDP finito, es decir, con acciones y estados finitos,es muy común representar estos valores según el cuadro 3.1, la cuál es denominada comúnmente como *tabla q*:

Cuadro 3.1: Tabla de relación estado-acción

	a_1	a_2	a_3	...	a_n
s_1	$q_{1,1}$	$q_{1,2}$	$q_{1,3}$	...	$q_{1,n}$
s_2	$q_{2,1}$	$q_{2,2}$	$q_{2,3}$	...	$q_{2,n}$
s_3	$q_{3,1}$	$q_{3,2}$	$q_{3,3}$	...	$q_{3,n}$
.	:	:	:		.
s_m	$q_{m,1}$	$q_{m,2}$	$q_{m,3}$	...	$q_{m,n}$

3.3.1 Aprendiendo las funciones de valor

Existen una gran variedad algoritmos para poder aprender los las funciones de valor o funciones de valor-acción; los cuales podrían clasificarse en los 3 grupos

siguientes:

- **Programación Dinámica (PD):** Se refiere a un conjunto de algoritmos que necesitan un modelo perfecto del ambiente para poder calcular la pol´ıtica óptima usando los valores de funciones. Dentro de ellos se encuentran los algoritmos de iteración del valor, iteración de la política, entre otros.
- **Monte Carlo (MC):** Al contrario de PD no requiere el conocimiento exacto del ambiente, el ambiente solo necesita generar transiciones.
- **TD Learning (TD):** Los métodos de diferencia temporal o *TD Learning*, se caracterizan porque como en MC, los agentes aprenden de la experiencia sin tener un modelo completo de el ambiente, y como en PD estos actualizan los valores estimados basados en parte de otros estimados, sin esperar a que se llegue a un estado terminal con un valor de salida. Algoritmos como Sarsa o Q-Learning son los más representativos.

Q-L EARNING

Dentro de los algoritmos de diferencia temporal, se tiene una ramificación: *on-policy* y *off-policy*. Q-Learning es un algoritmo *off-policy*, lo que significa que la función acción-valor aproxima directamente de la acción óptima del siguiente estado s'. En su forma más simple se puede definir como la ecuación 3.3:

$$Q(s_t, a_t) \longleftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)) \tag{3.3}$$

donde γ es la tasa de descuento de la función valor-acción del estado siguiente s_{t+1}.

3.3.2 ESTRATEGIAS DE SELECCIÓN DE ACCIONES

Uno de los principales dilemas en aprendizaje por refuerzo, es el de *exploración* vs *explotación*. Para obtener una recompensa máxima el agente, estará tentado a tomar la acción con el mayor valor que tenga en su conocimiento actual, sin posiblemente haber *explorado* todas las opciones, y tener conocimiento del valor de otras acciones. Para esto se tienen diferentes estrategias y las principales son las siguientes:

- **Greedy:** También conocida como golosa, el agente toma la acción que tenga el mayor función de valor o acción-valor con una probabilidad de 1.
- **E-greedy:** Se podría decir que es la evolución de la estrategia golosa. La acción que tenga el mayor valor o acción-valor tendrá una probabilidad de $1 - E$

de ser tomada, mientras que el resto de las acciones tendrán una cantidad equiprobable de ser escogidas, es decir, $\frac{E}{n-1}$, donde n es el número de acciones posibles a tomar.

- **Softmax:** Al contrario de $E-greedy$, las probabilidades de todas las acciones, dependen directamente de la funciones de valor-acción.

SOFTMAX

En el modelo propuesto se utilizó la estrategia de selección de acciones llamada softmax. El agente, al tener el conjunto de funciones de valor o funciones de valor-acción en el estado actual s, para cada acción; genera una distribución de probabilidad. Es decir, la probabilidad de que tome la acción a en el estado s está dada por la ecuación 3.4:

$$\pi(s,a) = \frac{e^{Q(a)/\tau}}{\sum_{i=1}^{n} e^{Q(i)/\tau}} \tag{3.4}$$

Donde n es el número total de acciones en el estado s, e es el exponente del logaritmo natural, y τ es un parámetro positivo llamado *temperatura*. Altas temperaturas causan que las probabilidades de las acciones sean casi equiprobables, mientras que bajas temperaturas causan una diferencia más grande de probabilidades. Cuando se llega al límite de que $\tau \longrightarrow 0$, se convierte en una estrategia totalmente golosa o *greedy*.

3.4 SISTEMAS MULTI-AGENTE

Un sistema multi-agente (MAS) puede definirse como el conjunto de agentes interactuando en un determinado ambiente, sea cooperando y/o compitiendo para la solución de un determinado problema. En el caso de que este sistema esté desarrollado para controlar hardware (robots), se lo puede denominar como sistema multi-robot (MRS) [14].

3.4.1 TEORÍA DE JUEGOS

El estudio de la toma decisiones en ambientes interactivos, basándose en matemática, estadística, ingeniería, economía, investigación operativa entre otras disciplinas; se llama teoría de juegos. Las interacciones entre agentes o individuos, son mode-

lados para poder analizarlos y optimizar las decisiones para que la recompensa obtenida sea mayor.

Como punto de partida, se puede tomar a los juegos *estocásticos*, donde el juego se desarrolla en una secuencia de etapas. Empiezan en un estado, y los individuos toman decisiones, cambiando posiblemente su entorno, recibiendo una recompensa por la acción tomada, y encontrándose en un estado distinto.Cuando el juego involucra la existencia de un único estado, se llama juego *repetitivo*. Ejemplos de juegos repetitivos es yan-quen-po, dado un estado, ejecutando una acción determinada termina el juego.

Para el caso de la propuesta de tesis se modela el ambiente como un juego repetitivo. Por esta razón es que la formula para la actualización de cualquier valor sea de función o estado, se toma como que el tiempo no tiene al infinito, sino que es un tiempo t, sin existir un $t + 1$.

3.4.2 Aprendizaje por Refuerzo con Valores de Influencia

En [2] se propone un nuevo algoritmo para la comunicación entre agentes en un entorno multi-agente. Se inspira en las interacciones sociales de las personas en una comunidad. La figura 3.2 muestra la interacción entre agentes.

Vemos que los agentes B y C puede elogiar o protestar una acción tomada de por el agente A. Esta opinión la definiremos como la opinión de los agentes OP . En base a esta opinión, el agente A, recibe una opinión del la comunidad, que la nombraremos el valor de influencia, o Influence Value IV .

$$IV = \sum_{j \in [i:N] \neg i} \beta_i(j) * OP_j(i) \tag{3.5}$$

donde $\beta_i(j)$ es la influencia del agente j sobre el agente i, $OP_j(i)$ es la opinión del agente j en relación a las acciones ejecutadas por el agente i. Es importante notar que el coeficiente de influencia B determinará si un agente será o no influenciado por la opinión de los agentes.

La opinión que los agentes tienen, será hecha a partir de la recompensa que obtuvieron del ambiente es así que:

$$OP_j(i) = \begin{cases} RV_j * OI(s_j(t), a_i(t)) & \text{si } RV_j < 0 \\ RV_j * (1 - OI(s_j(t), a_i(t))) & \text{si } RV_j > 0 \\ 0 & \text{Caso contrario} \end{cases} \tag{3.6}$$

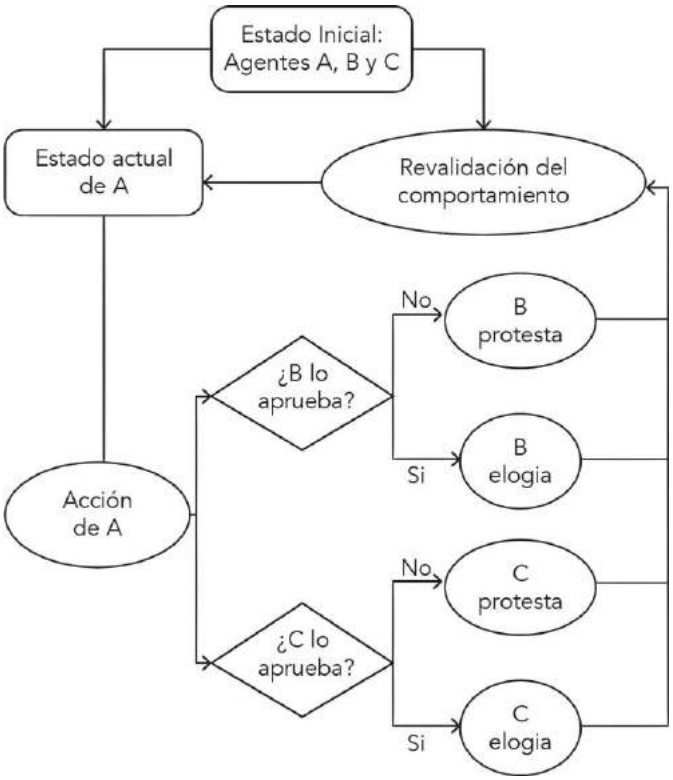


Figura 3.2: Comunicación entre agentes en un entorno multi-agente [14]

donde RV_j representa la referencia base de las opiniones del agente definida por:

$$RV_j = r_j(t+1) + V(s_j(t+1)) - V(s_j(t)) \tag{3.7}$$

y donde $OI(s_j(t), a_i(t))$ es el ´ındice de ocurrencia. El mismo es calculado como el porcentaje del nú mero de veces en que un agente i ejecuta una acción a_i en un estado de juego en el tiempo (t).

En el algoritmo Q-Learning, el valor Q se actualiza de la siguiente manera:

$$Q(s(t), a_i(t)) \longleftarrow Q(s(t), a_i(t)) + \alpha(r(t+1) + \gamma \max_a Q(s(t+1), a) - Q(s)t, a(t))) \tag{3.8}$$

Tomando como referencia el algoritmo Q-Learning, al introducir los valores de influencia genera el algoritmo IVQ-Learning, cuyo valor Q se actualiza de la siguiente forma:

$$Q(s(t), a_i(t)) \longleftarrow Q(s(t), a_i(t)) + \alpha(r(t+1) + \gamma \max_{a_i \in A_i} Q(s(t+1), a_i) - Q(s(t), a_i(t)) + IV_i) \tag{3.9}$$

donde $Q(s(t), a_i(t))$ es el valor de la acción $a_i(t)$ ejecutada por el agente i, α es el coeficiente de aprendizaje $(0 \leq \alpha \leq 0)$, γ es el coeficiente de descuento $(0 \leq \gamma \leq 0)$, $r_i(t+1)$ es la recompensa obtenida por el agente i y IV_i es el valor de influencia del agente i.

La ecuación 3.9 se usa en el capitulo 5 para describir el algoritmo utilizado en el modelo continuo del agente neuronal.

4 TRABAJOS RELACIONADOS

El estudio de redes neuronales artificiales, es uno de los más grandes en el área de computación hoy en día. La primera publicación en neurociencia [37], dió como resultado el estudio de redes neuronales aproximadamente hace medio siglo. Este primer modelo matemático fue inspirado en la neurona biológica; y actualmente es aplicada a una gran variedad de campos de acción [12]:

- Análisis de imágenes adquiridas satelitalmente;
- Clasificación de patrones en sonido e imágenes;
- Reconocimiento de rostros con visión computacional;
- Control de trenes a alta velocidad;
- Predicción de valores en el mercado financiero;
- Identificación de anomalías en imágenes médicas;
- Identificación automática de perfiles de clientes para empresas financieras;
- Control de aparatos electrónicos como máquinas de lavado, microondas, refrigeradores, máquinas de café, videocámaras, entre otros.

Pero no siempre fue así, el primer modelo solamente solucionada problemas básicos que sean linealmente separables, haciendo su campo de acción bastante reducido. Conforme pasaban los años, nuevos modelos han sido creados, tanto inspirados en la biología como solamente inspirados matemáticamente para un mejor desempeño del problema a resolver. Uno de los primeros problemas básicos, que aún no pudo resolver el primero modelo, fue la operación lógica XOR, problema que fue resuelto con la llegada de la ”segunda generación”de redes neuronales. Actualmente, existe una gran variedad de redes neuronales, cada una de ella creada con un propósito distinto a la otra, e inclusive pequeñas modificaciones de ellas para poder resolver un problema en particular de una manera única que mejore el

desempeño en comparación de la aplicación de la red en su forma básica o en la forma que fue creada. Vemos que resuelve la clasificación de semi-conductores en la industria, clasificación de cúspides dentales en el área de odontología, la detección de anormalidades mamarias en mamografías digitales en el área de medicina, y la aplicación en diferentes áreas como energía, textiles, alimentaria, hacen notoria nuevamente la importancia que las redes neuronales tienen en la actualidad.

4.1 Redes neuronales

Se han creado muchos modelos, definidos en diferentes niveles de abstracción y modelando diferentes aspectos de los sistemas neuronales. El primer modelo de red fue hecho en 1943 con Warren MacCulloch y Walter Pitts, un neuro-fisiólogo y matemático respectivamente [37]; estas redes neuronales fueron creadas con una salida binaria con simples funciones lógicas basándose en características de la neurona biológica; este primer modelo tiene el nombre de perceptrón. Luego, con el descubrimiento del backpropagation [48] y cambiando la neurona linear por una de función sigmoidal [11] se dio pase a la segunda generación de redes neuronales, de donde se desprenden modelos tan conocidos como las redes recurrentes [29], redes de la teor´ıa de la resonancia adaptativa (*ART*) [22], redes neuronales convolucionales [34], entre muchos aún posteriores a estos. Según la clasificación generacional los modelos antes mencionados están clasificados dentro de las redes neuronales de segunda generación, y el inicio de estos da fin también a la llamada época del invierno de las redes neuronales donde se perdió mucho interés en estas. Estos modelos tienen como característica principal una función de activación con un conjunto de posibilidades de salidas continuas. Sin embargo años mas tarde en [36] se desarrolló un nuevo modelo de red llamado *red neuronal de impulsos* o SNN(*spiking neural network*), donde cada neurona, inspirada en las neuronas biológicas, usa potenciales de acción para disparar información codificada, donde una variable principal es el tiempo, es decir, las neuronas no disparan con cada muestra. Estas redes SNN son la tercera generación de redes neuronales, y tienen muchas características conocidas del cerebro humano [59], es decir, se les considera las que más se asemejan a su funcionamiento. La clasificación generacional se encuentra en la literatura en los artículos relacionados a las redes neuronales de impulsos, pero no se encontró tal clasificación en publicaciones de otros tipos de redes, sin embargo es importante tal clasificación desde un punto de vista neurocientífico.

La clasificación más conocida es por los tipos de problemas a resolver; sean redes para aprendizaje supervisado, donde se infiere una función de data de entrenamiento etiquetada, o para aprendizaje no supervisado, donde se trata data no etiquetada. Los principales problemas que se enfrentan con aprendizaje supervisado son:

- **Aproximación de funciones:** El objetivo es encontrar una función que se adecue a la data de entrenamiento, la particularidad es que generalmente la variable de salida es continua.
- **Clasificación:** Dado un patrón de entrada de una de las clases ya definidas, asociar un patrón desconocido con su clase respectiva.

Y el principal problema para aprendizaje no supervisado es:

- **Clustering:** El objetivo es encontrar similaridades y particularidades de los patrones de entrada para permitir el agrupamiento de ellos.

Para poder crear los modelos neuronales hasta la actualidad, se siguieron dos vertientes para la sustentación de la creación de un nuevo modelo. La primera vertiente es la inspiración biológica donde los científicos inspirados en la funcionalidad del cerebro crearon redes artificiales con ciertas caracter´ısticas similares a este; y la segunda vertiente es por la capacidad computacional de los nuevos modelos creados, donde los modelos fueron creados para mejorar la precisión de algoritmos, o resolver nuevos tipos de problemas presentados. Dada la gran cantidad de modelos y aplicaciones, es necesario realizar un análisis más profundo de los modelos neuronales básicos desde su inspiración biológica y capacidad computacional.

4.2 ANÁLISIS DESDE LA INSPIRACIÓN BIOLÓGICA

El éxito de las redes neuronales artificiales, fue gracias a la inspiración biológica del primer modelo, dado que el cerebro provee una prueba de que la inteligencia es posible, los cient´ıficos usaron ingenier´ıa inversa a los principios computacionales detrás de la funcionalidad del cerebro, y duplicaron ciertas características de el, inspiración de donde salieron muchos modelos usados en la industria en la actualidad. Para una mejor compresión es necesaria una introducción a las características de las redes neuronales biológicas.

4.2.1 Red Neuronal Biológica

La red neuronal biológica esta formada por la interconexión de millones de neuronas, y se estima que cada una de estas neuronas tiene entre 10^3 y 10^4 de conexiones con otras neuronas. La neurona, como célula fundamental del sistema nervioso, tiene como rol principal conducir impulsos (estímulos eléctricos originados de reacciones f´ısico-qu´ımicas). Estos impulsos son realizados cuando las neuronas pre-sinápticas alcanzan el umbral de su potencial de acción y al ser transmitido este a través del axón, se liberan neurotransmisores que dependiendo del tipo de molécula producen la llamada sinapsis neuronal, inhibitoria o excitatoria según sea el caso, con las neuronas postsinápticas. Una sinapsis excitatoria aumenta la probabilidad de realizar un impulso de la neurona postsináptica, mientras que una sinapsis inhibitoria la disminuye.

La transmisión de señal entre neuronas es el núcleo principal en las capacidades de procesamiento del cerebro. Hay teorías en que la base de la asociación, memoria, y muchas otras habilidades mentales son gracias a la plasticidad sináptica, que es la capacidad de la sinapsis de ser más débil o fuerte según los incrementos o decrementos en su actividad, y es una de la claves para los modelos neuronales artificiales más conocidos.

Otro concepto importante es el mapeo topográfico neuronal, en donde cada área de nuestro cerebro es arreglada topográficamente, como lo muestra la imagen 4.1, donde se distinguen áreas específicas para distintas funcionalidades del cerebro. Por ejemplo, las neuronas en el área visual de la corteza son arregladas topológicamente, en el sentido que neuronas adyacentes tienen campos receptivos visuales que constituyen un mapa de la retina. Y es aqu´ı donde se puede empezar a hablar de una auto-organización de las neuronas, dada las billones de conexiones de neuronas y sus miles de conexiones, estas se ordenan de una forma que posibilitan acciones macro tan complejas como la inducción.

La mayoría de redes neuronales tienes las siguientes características en común con el sistema nervioso:

- El procesamiento de información ocurre en muchos elementos simples llamadas neuronas.
- Estas neuronas pueden recibir y enviar est´ımulos a otras neuronas
- Las neuronas están interconectadas formando redes neuronales.

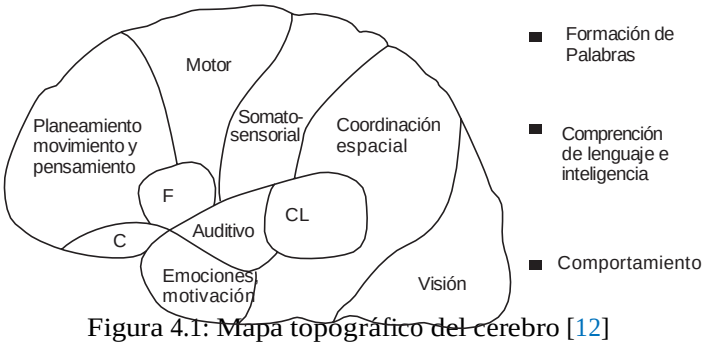


Figura 4.1: Mapa topográfico del cerebro [12]

- La información (señales) es transmitida entre neuronas por medio de conexiones llamadas sinapsis.
- La eficiencia de la sinapsis, representada por el valor del pesos asociado, corresponde a la información almacenada.
- El conocimiento es adquirido del ambiente por un proceso de aprendizaje, que básicamente responsable de adaptar la fuerza de las conexiones al estimulo del ambiente.

Pero cada modelo tiene ciertas caracter´ısticas particulares, y otros no. La neurona que Mc-Cullock y Pitts formularon es del tipo ”todo o nada”, es decir, una forma binaria de comunicación. Sin embargo el conocimiento actual que tenemos, sabemos que los procesos eléctricos y químicos de la neurona no es solo una proposición lógica dada la cantidad de neurotransmisores que pueden transmitir información entre neuronas. Los creadores tomaron en cuenta las siguientes suposiciones biológicas:

- El comportamiento de la neurona es un proceso binario;
- En cualquier tiempo, un número de sinapsis deben ser excitadas para activar a la neurona.
- La excitación de una neurona puede ser inhibida por una sinapsis inhibitoria.
- La red neuronal tiene una estructura estática, es decir, su estructura no cambia en el tiempo.

Para la creación del perceptrón multicapa, fue necesario el descubrimiento de la retropropagación, algoritmo que usan muchas redes neuronales, incluyendo las

convolucionales para poder aprender. La retropropagación puede que sea una de los algoritmos más difíciles de inspirar biológicamente, dado que se necesitaría también que las neuronas post-sinápticas transmitan a través de las dendritas información a los axones de las neuronas pre-sinápticas y estas sucesivamente hacia atrás. A pesar de la dificultad, se puede reescribir la ecuación de la retropopagación de forma que no se ignore las características biológicas.

Otra modelo importante son las redes recurrentes de Hopfield. La red neuronal recurrente a pesar que puede resolver varios tipos de problemas, especialmente los que tengan que ver con series de tiempo, no tuvieron una inspiración biológica para su creación. Hopfield se dio cuenta que varios sistemas que tienen un comportamiento espontáneo, se tener una memoria asociativa para recuperar un patrón almacenado como respuesta a una presentación incompleta o con ruido del patrón. Esta red también ayudó a que el invierno de las redes neuronales artificiales terminara.

Las redes llamadas de teor´ıa de resonancia adaptativa, usa una memoria a corto y largo plazo (STM, LTM). Los creadores no asocian su red a una inspiración biológica, sin embargo se puede asociar estas redes con las caracter´ısticas de memoria que tiene nuestro cerebro:

- **Short Term Memory:** Dura de unos segundos a unos pocos minutos.
- **Intermediate long-term memory:** Dura de minutos a semanas.
- **Long-term memory:** Dura por un tiempo indefinido de tiempo

Los mapas auto-organizativos de Kohonen, formula una teor´ıa que a pesar de que ellos mismos intentan no afirmar la auto-organización del cerebro dado que en aquel tiempo se evidenciaba más un factor genético que el factor auto-organizativo; se inspiran en la auto-organización.

Las redes convolucionales tienen una inspiración biológica en la corteza cerebral, particularmente en las células responsables de la selectividad de orientación y detección de bordes dentro de corteza visual primaria V1.

Las redes de impulsos, al ser las más parecidas al cerebro tiene bastantes caracter´ısticas similares. Empezando por el nombre, las neuronas al tener un umbral de potencial de acción no liberan energía en cada muestra como sucede en las redes anteriormente citadas, sino que estas pueden o no dar impulsos en el tiempo. Las redes artificiales de impulsos igualmente funcionan con un umbral de potencial de acción. Además también tienen impulsos inhibitorios y excitatorios, y también

pesos que representan la plasticidad sináptica cerebral. Sin embargo estas redes, a pesar de su parecido a las redes neuronales biológicas, no llegan a la precisión del estado del arte en los problemas comunes.

4.2.2 AUTO-ORGANIZACIÓN

La auto-organización fue introducida en las redes neuronales gracias a los mapas auto-organizativos de Kohonen [31]. La auto-organización está presente en un amplio rango de procesos y formación de patrones en sistemas físicos y biológicos, como las aves en completa formación al volar. Lo interesante de este fenómeno es que los sistemas auto-organizados son generalmente aislados de las leyes f´ısicas o reglas globales. Algunas características de la auto-organización son:

- Usualmente son compuestos por un número largo de elementos que interactúan entre ellos y el ambiente;
- La multiplicidad de las interacciones enfatiza que es un sistema dinámico y requiere interacciones continuas;
- Usualmente se nota que aparece cierto orden espontáneo;
- La interacción de los componentes no se puede entender como una interacción lineal, como la simple adición de contribuciones individuales;
- La mayor´ıa de sistemas auto-organizados son complejos.

La auto-organización, en el cerebro sin embargo, al contrario de lo que se sabía en 1982, donde se creó los mapas auto-organizados, ahora se tienen más pruebas de la auto-organización neuronal, dado que las neuronas en el cerebro son billones y tienen miles de conexiones cada una, no se puede tener tanta información genética para determinar la estructura de estas, sino que su estructura esta gobernada por el fenómeno de auto-organización.

La primera generación de redes neuronales basadas en la neurona de *McCulloch-Pitts*, fueron motivadas en el conocimiento de su época de una neurona como unidad de umbral lógica, dando solo salidas binarias de comunicación, resolviendo cualquier tipo de función booleana, comúnmente se llama modelo de ”todo o nada”, formulado con las siguiente 5 suposiciones lógicas.

- El comportamiento de la neurona es un proceso binario;

- En cualquier tiempo, un número de sinápsis

Esta primera generación, duró por aproximada mente 20 años.Luego la segunda generación de redes neuronales son basadas en unidades computacionales que aplican una función de activación con un conjunto de salidas continuas de salida. Esta generación evolucionó gracias al algoritmo de *backpropagation* [48] y el cambio de neurona linear con una signoidal [11]. Interpretando biológicamente, las neuronas de la segunda generación ve la salida de una unidad signoidal como la velocidad de disparo de una neurona biológica, ya que que estas disparan a frecuencias intermedias, entre sus frecuencia mínima y máxima. Con esta interpretación de *velocidad de disparo* las redes de segunda generación son biológicamente más realistas que las de la primera generación.
Dentro de esta generación, fue creado el modelo de mapas auto-organizativos, inspirados biológicamente en la auto-organización biológica. Sin embargo importante notar que las neuronas que forman el mapa de auto-organización no son ”neuronas”perse, sino que estas forman una especie de clustering, y no son una unidad de procesamiento como lo sucede en un perceptrón.

Sin embargo en el campo de neurociencia, gracias al trabajo publicado en [28] describiendo como los potenciales de acción de las neuronas son iniciadas y propagadas, y la evidencia experimental indicando que los sistemas biológicos neuronales usan el tiempo del potencial de acción para codificar la información neuronal; contribuyo a la creación de la tercera generación de redes neuronales que usa neuronas de impulso como unidades computacionales [36].

Las neuronas de impulso asumen que una neurona dispara cuando su potencial de acción alcanza un umbral determinado, donde este potencial de acción es la suma de potenciales excitatorios postsinápticos (EPSPs) y los potenciales inhibidores postsinápticos (IPSPs) de neuronales presinápticas. Una característica fundamental de este modelo es el tiempo, dado que las neuronas no disparan cada vez que entra data a la red, sino mas bien cuando su potencial de acción llegó al umbral. A pesar de la semejanza biológica como se muestra en [20] las neuronas de impulso aún son poco investigadas dado que toman mucho tiempo para entrenar.

4.3 Análisis desde su capacidad computacional

Sin embargo es aún más interesante estudiar los modelos desde la clasificación común de aprendizaje supervisado y no supervisado, dado que esta clasificación

tiene un enfoque en los tipos de problemas a resolver, más que en la inspiración biológica.

Uno de los argumentos que no tienen como guía para la creación de nuevos modelos a la inspiración biológica, es que no se tiene aún suficiente conocimiento del cerebro para poder usarlo como guía.

Cuadro 4.1: Tabla de doble entrada comparativa.

	Clasificación	**Regresión**	**Clustering**
Feed Forward	[40] Speech Recognition	[17] Reinforcement Learning	
Convolutional Neural Network	[13] Speech Recognition	[58] Counting Cells in Microscopy	
	[15] Object Detection		
	[26] Object Recognition		
Recurrent Neural Network	[49] Speech Recognition	[44] Predicting Stocks	
	[24] Text Classification		
Adaptive Resonance Theory	[18] Control of Hands Movements		
SOM y GNG	[15] Object Detection		[55] Speech Recognition
	[9] Image Recognition		[1] Image Detection

Continua en la siguiente página

Cuadro 4.1 – Continuación de la anterior página

	Clasificación	**Regresión**	**Clustering**
	[33] Image Recognition		
Spiking Neural Network	[35] Feature Extraction on Speech	[45] Finantial Prediction	
	[43] Object Recognition		

Además de su importancia en las aplicaciones cotidianas, vemos que los modelos fueron evolucionando para resolver diversos problemas computacionales que no entran en la , por ejemplo en [10] se crea un nuevo modelo neuronal para aprender automáticamente las estructuras de redes neuronales con un desempeño competitivo; en [30] se crea una red neuronal que supera la limitación del olvido catastrófico donde cada peso sináptico no solo almacena su valor, sino también una representación de incertidumbre acerca del peso, disminuyendo la probabilidad de cambio de los pesos de las tareas ya aprendidas; en [19] se crea una red llamada *Pathnet* para resolver la deficiencia de transferencia del aprendizaje de las redes neuronales, es decir, donde se transfiere el conocimiento de una tarea aprendida A hacia una nueva tarea por aprender B, acelerando el proceso de aprendizaje de la nueva tarea. En [23] se crea una red nueva llamada *OCN*, está basada en redes convolucionales, pero para lidiar con el problema de estabilidad-plasticidad, se crean dos *OCN* con diferentes reglas de actualización, creando así un framework ensamblado, la aplicación es en tracking de videos online.

El uso de las redes neuronales para aprendizaje supervisado se puede resumir en interpolación de funciones y clasificación. Vemos por ejemplo que se usan las redes neuronales para casos espec´ıficos de algoritmos donde se necesita interpolación en partes específicas de algoritmos como Optical Flow, [60] donde se usa las redes convolucionales para la fase de interpolación, esto es inspirado biológicamente por el fenómeno ”perceptual filling”, que es el rellenado de color, texturas, brillo, movimiento ante diferentes imágenes incompletas presentadas al cerebro.

En la actualidad se usa mucho el deep learning, por ejemplo vemos sus usos en el alineamiento de rostros [32] (donde se pretende mapear los principales rasgos de un rostro como la boca, nariz, ojos), se usa una red convolucional de múltiples escenarios, donde cada en cada uno se hacen mapas de calor, para que en el siguiente escenario mejore la precisión de los puntos de referencia clave.

Sin embargo, aunque se usan las redes convolucionales solas, se puede ver que pueden combinarse con diferentes tipos de redes, para poder crear algoritmos más completos, por ejemplo para el problema de emparejamiento geométrico (donde se transforma la imagen A con respecto a un emparejamiento de la imagen B) se usan dos redes idénticas para las dos imágenes a emparejar [47], luego se hace una red para emparejamiento y luego la red para la regresión. Otro ejemplo es al combinarlas con una red neuronal recurrente [52] se hace una red *end to end* donde la primera capa es una CNN y luego una LSTM bidireccional para poder predecir una secuencia de texto.

A pesar de su gran impacto en el desempeño de muchos problemas, no solo se usa las redes convolucionales. Para solucionar un problema parecido al anterior de etiquetado de secuencias, [16] se crea una mejora a la red recurrente que mejora a las redes LSTM y GRU específicamente en ese aspecto, donde a diferencia de las anteriores redes recurrentes, se embebe etiquetas lo que mejora el rendimiento de esta red; además otra mejora es que no tiene tantos parámetros como LSTM.

En el mismo campo de procesamiento de lenguaje natural, se ve la necesidad de una memoria a largo plazo dado que las memorias de las redes recurrentes son pequeñas, es preciso una memoria a largo plazo que después de procesar un texto pueda responder preguntas de este por lo que en [56] se crea un nuevo modelo de red que basado en estrategias de aprendizaje máquina de inferencia, tenga una memoria que pueda ser leída y escrita. Otro nuevo modelo que se crea con bases matemáticas es [10] dada la gran cantidad de parámetros que pueden tener las redes, se creó el nuevo modelo para que pueda aprender la estructura y parámetros de las redes, reduciendo así el tiempo de búsqueda de estas variables. Otro problema que están retomando a tomar interés los científicos, es el problema de aprendizaje continuo, donde se dan cuenta que es necesaria una red que interpole correctamente varios problemas a la vez. Para las redes neuronales de impulsos, se creó un algoritmo [42] que pueda olvidar para que pueda aprender nuevos problemas, haciendo un balance olvido-aprendizaje bueno.

4.4 Aprendizaje por Refuerzo

Los algoritmos de aprendizaje por refuerzo conocidos como Sarsa, Monte Carlo, Q-Learning, entre otros; fueron concebidos para desempeñarse en un MDP puro. Sin embargo, muchos problemas reales, requieren trabajar en ambientes donde el espacio de estados o acciones es continuo. Para esto, las soluciones creadas, se pueden clasificar en 3 enfoques [54].

Aproximación del modelo

Los modelos por aproximacion de modelos [41],calculan la politica deseada aproximando el MDP. Se intenta aproximar las funciones de transicion y recompensa, mientras que los estados, acciones y el factor de descuento son conocidos.

Aproximación del valor

Los algoritmos de aproximación del valor usan la experiencia del agente para actualizar el valor de estado $v*$o el valor del par estado-accion $q*$. Muchos algoritmos son de esta rama, por esto, la clasificación mas conocida es *On-policy* y *Off-Policy*.

Finalmente, estos algoritmos terminan almacenando una función aproximada para el estado, una función aproximada para las acciones o las dos a la vez.

Aproximación de la política

Los algoritmos que aproximan la politica, escogen una pol´ıtica base, y a partir de esta intentan encontrar la politica optima. Los algoritmos que almacenan solo la politica son llamados *actor-only*, mientras que los que almacenan la politica y la funcion de valor son llamados *actor-critic*.

4.4.1 Aproximación de funciones

Sin embargo para cualquiera de estas técnicas, se necesitan formas de almacenar y actualizar una funcion aproximada. Estas técnicas se pueden clasificar en aproximadores de funciones lineales y no lineales.

Aproximación de funciones lineales

Los aproximadores lineales, son útiles porque su complejidad es menor que los aproximadores no lineales, además que son más rápidos de procesar, sin embargo depende criticamente en como los estados son representados en caracter´ısticas.

Además de la aproximación lineal usando un vector de parámetros, y ajustando estos al estido de regresión lineal, una técnica importante y la más intuitiva de hacerlo, es discretizando el espacio. *Coarse coding* [27] es una de las primeras discretizaciones existentes, donde el estado actual es un punto en un espacio d-dimensional, el cual deberá caer en un nú,ero limitado de subespacios que se intersecan, sin embargo, no es una tan usada como *Tile coding*. *Tile coding*, es una de las técnicas más usadas para discretizar el estado *[51]*, también llamada CMAC, es una forma de *coarse coding* donde una de las ventajas .

Sin embargo, las aproximaciones lineales no lidian muy bien con espacios de muchas dimensiones, principalmente porque no generalizan bien. Discretizar el espacio, tampoco es una buena alternativa porque el numero de estados crece exponencialmente con respecto del número de dimensiones del problema, por esto se vuelve más conveniente las aproximaciones no lineales.

Aproximación de funciones no lineales

Una de las tecnicas mas conocidas en las aproximaciones lineales, es el uso de redes neuronales. Por ejemplo [39], se hizo bastante conocido porque se usaron redes neuronales convolucionales para la aproximacion de funciones en RL con mucho exito, probandolo en diferentes juegos de Atari, esto quiere decir con una gran cantidad de dimensiones. Otro caso exitoso es el uso de la técnica de actor-critic en [38], donde ya se cambia el uso de GPUs, por el uso de multiples procesadores, mejorando así la velocidad y rendimiento en comparación de agentes con redes convolucionales.

4.5 Consideraciones Finales

En la sección 4.2 hemos visto que las redes neuronales vienen desarrollándose con el tiempo con características semejantes a las redes neuronales biológicas como neuronas que disparan impulsos con el tiempo, redes que enfrentan el problema

de olvido catastrófico o aprendizaje continuo y redes que enfrentan el problema de transferencia de conocimiento. Analizamos a los modelos desde su capacidad computacional y desde su inpiración biológica, pero no se encontró un modelo auto-organizativo que tome a las neuronas como unidades de procesamiento.

5 PROPUESTA DE INVESTIGACIÓN

En este cap´ıtulo, presentamos el modelo neuronal propuesto de esta tesis. Como fue expuesto anteriormente, este modelo resolverá el problema de aprendizaje supervisado, que ya fue definido formalmente en el cap´ıtulo 2 espec´ıficamente el problema de clasificación. En adelante, llamaremos a la neurona de nuestro modelo, como agente neuronal.

5.1 MODELO PROPUESTO

5.1.1 NEURONA

Un agente neuronal, modelado matemáticamente es una función, pero al contrario de las neuronas inspiradas en la primera neurona de McCullow y Pitts, las cuales son funciones estáticas (Sigmoidales, RElu, etc.), esta función es variable. La figura 5.1 muestra su comportamiento, donde al modelarse como un agente de aprendizaje por refuerzo, tiene como entradas un estado $\boldsymbol{s}$, y tiene como salida la acciones a que el agente neuronal escoge según su política de selección de acciones π.

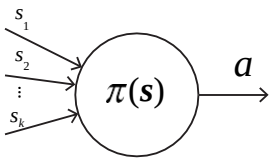


Figura 5.1: Neuron Model of IVQ-Net

La política π en la fase de entrenamiento se irá ajustando dependiendo de la estrategia de selección de la política escogida (softmax, e-greedy, etc.), mientras en la fase de prueba la estrategia sera completamente gulosa (greedy).

Cada agente neuronal se desempeña en un ambiente de juegos repetitivos, dado que las muestras $\boldsymbol{x}_1$ y $\boldsymbol{x}_2$, que a su vez se convierten en estados, no tienen correlación con respecto del tiempo. En los juegos repetitivos no se tiene la variable de

tiempo t, los estados iniciales son los estados terminales, por consecuencia la tasa de descuento es nula $\lambda = 0$. Por este motivo es que el algoritmo IVQ-Learning para juegos repetitivos, se simplifica en la ecuación 5.1. Se usará este algoritmo de aprendizaje por refuerzo como sistema de aprendizaje de los agentes neuronales.

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r - Q(s,a) + IV) \tag{5.1}$$

5.1.2 Discretización de los estados

Como comentamos en el capítulo 3, en RL es común discretizar el espacio de estados $\mathcal{S} \subseteq \mathbb{R}^k$ cuando este es continuo para poder simplificar el problema. Es nece- sario recalcar, que dado que los agentes están actuando en un ambiente de juegos repetitivos, los estados no tendrán una notación que dependa del tiempo.

Para discretizar el espacio de estados, primeramente es esencial normalizar la data $\mathcal{D}$. Luego, cada dimensión se debe de partir en un número de particiones b, dado esto, se tendrán b^k estados, donde k es el número de dimensiones del espacio de estados $\mathcal{S} \in \mathbb{R}^k$. Como ejemplo, en la figura 5.4, se tiene un espacio $\mathcal{S} \in \mathbb{R}^2$ de un numero de dimensiones $d = 2$, con un numero de particiones $b = 5$.

5.1.3 Red Neuronal

En cuanto a la red neuronal, estará conformada por L capas, donde cada capa es un conjunto de agentes, como lo muestra la figura 5.2. La entrada a la red, serán las muestras $\boldsymbol{x}^{(i)}$, mientras que la salida $\hat{\boldsymbol{y}}^{(\boldsymbol{i})} = \{a^L_1, a^L_2, .., a^L_m\}$ serán las acciones ejecutadas por las neuronas de la ultima capa $\boldsymbol{n^L}$.

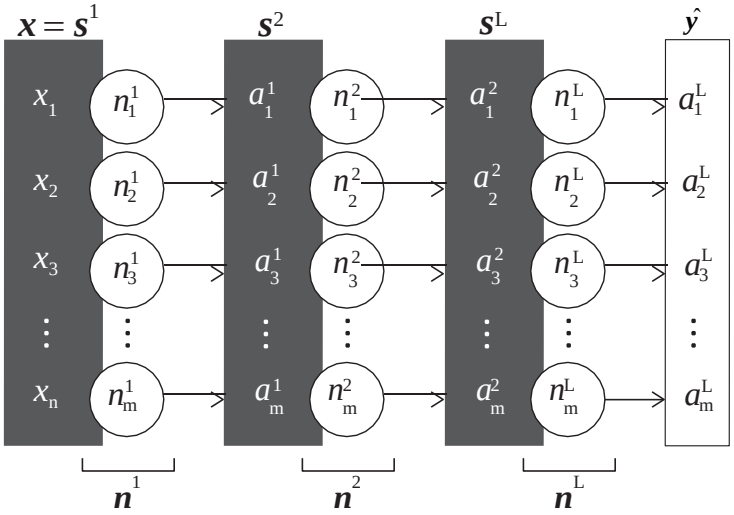


Figura 5.2: Fluido de la información hacia adelante de la red neuronal

Luego, para cada neurona en la capa $\boldsymbol{n^i}$, ejecuta su acción en el mismo estado $\boldsymbol{s}^i$, el cual es simplemente la acciones $\boldsymbol{a}^{i-1}$ de la capa anterior n^{i-1}. Para el caso del primer estado s^1, este sera igual a la muestra $\boldsymbol{x}$, la cual deberá ser discretizada según el procedimiento seguido en la sección 5.1.2.

Una vez, se tenga el vector de salida $\boldsymbol{\hat{y}(i)}$, se comparara la respuesta con el vector real de salida $\boldsymbol{y}$, es asi que se obtiene un vector $\boldsymbol{r} = \{r^1, r^2, ..., r^m\}$ de errores o recompensas, donde r_1 es el error cometido por la neurona n^L_1, y es calculado según la ecuación 5.2.

$$r_i = \hat{y}_i - y_i \tag{5.2}$$

Para el aprendizaje de la red, se requiere qeu cada agente pueda ajustar su valor Q usando el algoritmo de IVQ-Learning. Sin embargo, es necesario notar que los agentes de la ultima capa $\boldsymbol{n}^L$, son los únicos que tienen relación con el ambiente, por lo tanto los únicos que reciben recompensas. Los agentes de las capas restantes, recibirán una opinión, tal como lo muestra la figura 5.3.

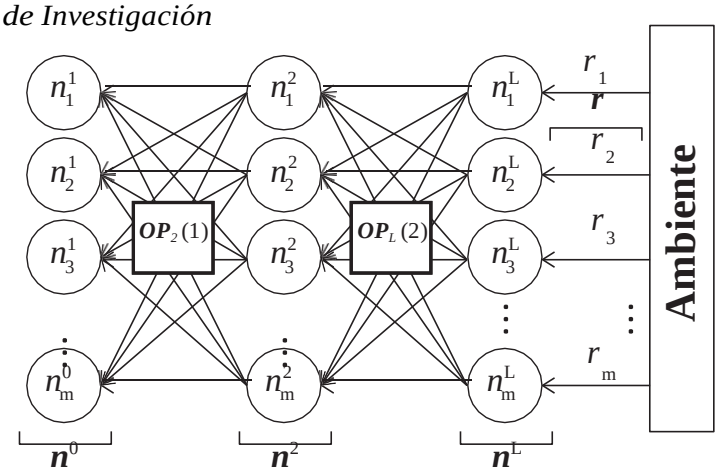


Figura 5.3: Fluido de la información hacia atrás la red neuronal

Por este motivo es necesario hacer una modificación al algoritmo original a la ecuación de calculo del valor de referencia. La nueva ecuación 5.3 es usada en el modelo.

$$RV_i \leftarrow \begin{cases} r_i - Q(s,a) & \text{Si la neurona está ne la última capa } \boldsymbol{n}^L \\ IV_i - Q(s,a) & \text{Caso contrario} \end{cases} \tag{5.3}$$

El algoritmo 1 muestra un resumen del aprendizaje de la red neuronal IVQ-Net.

5.2 Segunda Versión

La propagación del conocimiento de los estados, está inspirado en la forma en que las redes SOM influencian su conocimiento a las neuronas vecinas de la neurona ganadora, para esto se necesita discretizar el espacio de estados $\mathcal{S}$.

5.2.1 Propagación del conocimiento

Para esto, la tupla del estado actual se nota $\boldsymbol{s} = \{s_1, s_2, ..., s_k\}$. En la figura 5.4 el estado es una tupla donde $\boldsymbol{s}^{(i)} = \{0{,}5, 0{,}5\}$ el cual es el estado actual de la neurona $n^{(i)}$,representado en color negro; y los estados vecinos $\boldsymbol{s}^{(nei)}$ representados de color

gris son los estados influenciados por el conocimiento del estado principal $s^{(i)}$.

Ademas, para saber a cuantos estados se influenciará, se usará un radio de acción R, el cual definirá cuales son los estados vecinos, y por lo tanto, los estados a quienes se les influenciará el conocimiento. Este parámetro se escoge empíricamente dependiendo del problema, como lo analizaremos en el capítulo 6.

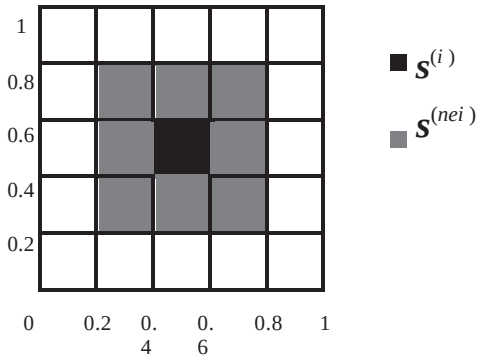


Figura 5.4: Espacio del estado con $k = 2$, $b = 5$, $R = 1{,}75$

El valor $Q(s^{(i)}, a)$ de los estados se calcula siguiendo la ecuación 5.1. Sin embargo, los estados vecinos que se encuentren dentro del radio R, seguirán la ecuación 5.4.

$$Q(s^{(nei)}, a) = Q(s^{(nei)}, a) + \alpha(\lambda \times RV_i) \tag{5.4}$$

Donde λ es el factor de propagación calculado según la ecuación 5.5 y donde s^{nei} es el estado vecino del estado principal s^i.

$$\lambda = \frac{-\|s^{(i)} - s^{(nei)}\|^2}{2\sigma^2} \tag{5.5}$$

Donde $s^{(i)}$ es el estado actual del agente i, $s^{(nei)}$ es estado el vecino, y donde RV es el valor de referencia del agente i.

El algoritmo 2 muestra el un resumen de algoritmo propuesto.

5.3 Intuiciones

Como se muestra la figura 5.5, cada neurona tendrá una función definida por la función Q, es decir, cada neurona tiene un conocimiento diferente a las de sus vecinas; además la diferencia más notable es que no se actualizarán los pesos sinápticos

como en las redes neuronales ANN, sino que las neuronas son las que aprenden y se organizan para lograr el objetivo.

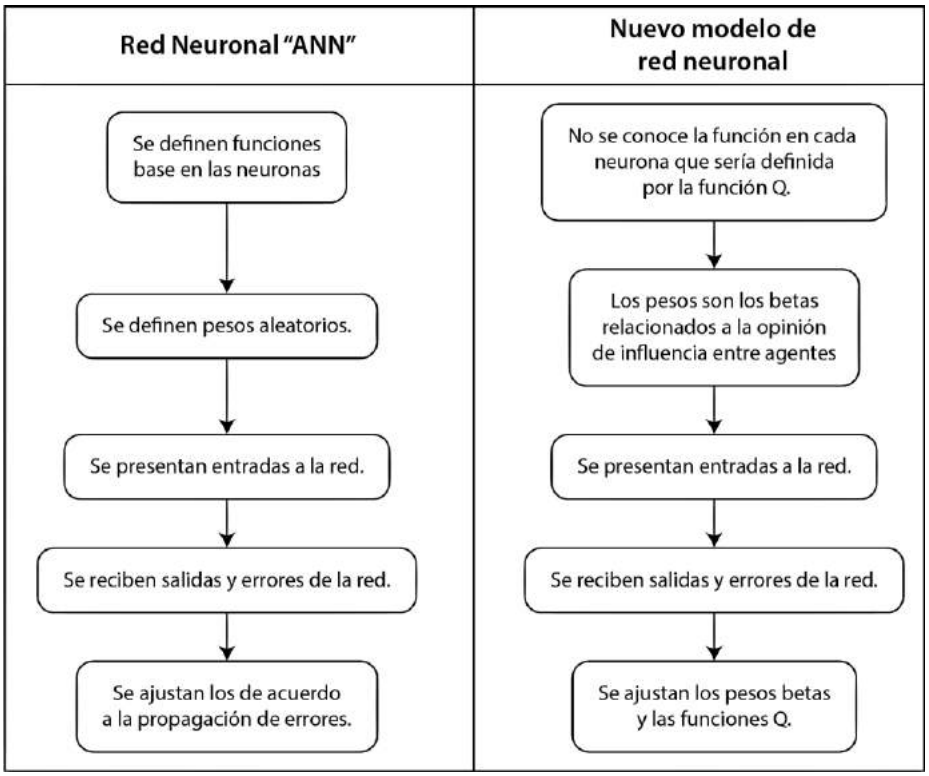


Figura 5.5: Proceso de entrenamiento de nuevo modelo de red neuronal

5.4 Métricas de Evaluación

Si bien el error, es de las métricas más básicas para definir el rendimiento de los modelos para aprendizaje supervisado. Este en varios casos no muestra la información suficiente para evaluar el desempeño del modelo, esto se ve especialmente reflejado en bases de datos con clases no balanceadas. En TODO, hacen un análisis de las métricas para clasificación. Dentro de ellas las métricas llamadas *recall* −*precision*, son de mucha ayuda para data no balanceada.

Para un mejor entendimiento de las métricas se necesitan definir los siguientes tipos de predicciones, los cuales se verán mejor reflejados en la la matriz de confusión 5.1.

- True positive (*tp*): Muestra clasificada como positiva y que realmente es positiva.
- False positive (*fp*): Muestra clasificada como positiva, pero que es negativa.
- True negative (*tn*): Muestra clasificada como negativa, pero que es positiva.
- False negative (*fn*): Muestra clasificada como negativa, y que realmente es negativa.

Base de datos	**Clasificado como positivo**	**Clasificado como negativo**
Positivo	True positive (*tp*)	False negative (*fn*)
Negativo	False positive (*fp*)	True negative (*tn*)

Cuadro 5.1: Matriz de confusión para base de datos binarias

Es importante recalcar, que en su forma básica, estas métricas son hechas para clasificación binaria. Sin embargo, estas se pueden extender para clasificación de múltiples clases. Para esto se tienen dos enfoques, micro-promedio y macro-promedio. Mientras que el enfoque macro-promedio trata a todas las clases por igual, el enfoque micro-promedio favorece a las clases más grandes. En este trabajo, se usará el enfoque macro.

- **Precision**: Es la fracción de instancias recuperadas que son relevantes. Mientras que en clasificación binaria está dada por la ecuación 5.6, mientras que para clasificación de múltiples clases está dada por la ecuación 5.7.

$$\frac{tp}{tp + fp} \tag{5.6}$$

$$\frac{\sum_{i=1}^{l} tp_i + fp_i}{l} \tag{5.7}$$

$$l \quad \underline{tp_i}$$

- **Recall**: Es la fracción de instancias relevantes que han sido recuperadas. Mientras que en clasificación binaria está dada por la ecuación 5.8, mientras que para clasificación de múltiples clases está dada por la ecuación 5.9.

$$\frac{tp}{tp + fn} \tag{5.8}$$

$$\frac{\sum_{i=1}^{l} tp_i + fn_i}{l} \tag{5.9}$$

$$\sum_{l}^{} \frac{tp_i}{}$$

- **Fscore**: Mientras que en las dos anteriores métricas existe diferencia dependiendo si es clasificación binaria o de múltiples clases, *Fscore* se mantiene igual, siguiendo la ecuación 5.10.

$$\frac{2 \times precision \times}{\underline{recall}\ precision + recall} \tag{5.10}$$

Algorithm 1 Algoritmo IVQ-Net

Require: Inicializar α, β, k

repeat
  Discretizar el vector de entrada $\boldsymbol{x}$
  for all capas $\boldsymbol{n}^i$, desde $\boldsymbol{n}^1$ hasta $\boldsymbol{n}^l$ **do**
    for all neuronas n_i^l en la capa $\boldsymbol{n}^l$ **do**
      Escoger la acción a_i^l en el estado $\boldsymbol{s}^l$, usando la política π
      Ejecutar a_i^l, observar r_i o IV_i
      Observar acciones de todas las neuronas de la capa $\boldsymbol{n}^{l-1}$
    end for
  end for
  for all neurona en $\boldsymbol{n}^L$ **do**
    $r_i^L \longleftarrow -/\hat{y} -$
  end for
  for all capas $\boldsymbol{n}^l$, desde $\boldsymbol{n}^L$ hasta $\boldsymbol{n}^1$ **do**
    for all neuronas n_i^l en la capa $\boldsymbol{n}^l$ **do**
      Calcular

$$RV_i \longleftarrow \begin{cases} r_i - Q(s,a) & \text{si la neurona está en la última capa } \boldsymbol{n}^L \\ IV_i - Q(s,a) & \text{caso contrario} \end{cases}$$

      Observar las opiniones de la capa $\boldsymbol{n}^{l+1}$

$$OP_j(i) \longleftarrow \begin{cases} RV_j * OI(s_j, a_i) & \text{si } RV_j < 0 \\ RV_j * (1 - OI(s_i, a_j)) & \text{si } RV_j > 0 \\ 0 & \text{caso contrario} \end{cases}$$

      Calcular

$$IV_i = \sum_{\substack{j \in 1:N- \\ i}} \beta_i(j) * OP_j(i)$$

$$Q(s,a) \longleftarrow Q(s,a) + \alpha(r_i - Q(s,a) + IV_i)$$

    end for
  end for
until convergencia

Algorithm 2 IVQ-Net versión 2

Require: Inicializar $\alpha, \beta,$ k, σ, R

repeat
  Discretizar el vector de entrada $\boldsymbol{x}$
  for all capas $\boldsymbol{n}^i$, desde $\boldsymbol{n}^1$ hasta $\boldsymbol{n}^l$ **do**
    for all neuronas n_i^l en la capa $\boldsymbol{n}^l$ **do**
      Escoger la acción a_i^l en el estado $\boldsymbol{s}^l$, usando la política π
      Ejecutar a_i^l, observar r_i o IV_i
      Observar acciones de todas las neuronas de la capa $\boldsymbol{n}^{l-1}$
    end for
  end for
  for all neurona en $\boldsymbol{n}^L$ **do**
    $r_i^L \longleftarrow -/\hat{y} -$
  end for
  for all capas $\boldsymbol{n}^l$, desde $\boldsymbol{n}^L$ hasta $\boldsymbol{n}^1$ **do**
    for all neuronas n_i^l en la capa $\boldsymbol{n}^l$ **do**
      Calcular

$$RV_i \longleftarrow \begin{cases} r_i - Q(s,a) & \text{si la neurona está en la última capa } \boldsymbol{n}^L \\ IV_i - Q(s,a) & \text{caso contrario} \end{cases}$$

      Observar las opiniones de la capa $\boldsymbol{n}^{l+1}$

$$OP_j(i) \leftarrow \begin{cases} RV_j * OI(s_j, a_i) & \text{si } RV_j < 0 \\ RV_j * (1 - OI(s_i, a_j)) & \text{si } RV_j > 0 \\ 0 & \text{caso contrario} \end{cases}$$

      Calcular

$$IV_i = \sum_{j \in 1:N-i} \beta_i(j) * OP_j(i)$$

$$Q(s^{(i)}, a) \longleftarrow Q(s,a) + \alpha(r_i - Q(s,a) + IV_i)$$

$$Q(s^{(nei)}, a) = Q(s^{nei}, a) + \alpha(\lambda \times RV_i)$$

    end for
  end for
until convergencia

6 Resultados experimentales

En este capítulo se describirán algunos ajustes necesarios para la validación del modelo propuesto en el capítulo 5, además de los data sets utilizados. Las pruebas fueron hechas en un procesador 2.3 Ghz. Intel Core i7, con 16 Gb de RAM y el lenguaje de programación usado fue C++

En este cap´ıtulo analizaremos los resultados del modelo propuesto.

- En la sección 6.1, hablaremos de algunos aspectos preliminares como las bases de datos usadas, y sus descripciones respectivas.
- En la sección 6.2 hablaremos del desempeño del modelo con las distintas bases de datos, comparándolo con el desempeño de un MLP.
- Finalmente, en la sección 6.3 analizaremos los parámetros usados en este modelo.

6.1 Aspectos Preliminares

Los algoritmos implementados en el presente trabajo, descritos en el cap´ıtulo 5, fueron testeados y validados con el objetivo de analizar el rendimiento y comportamiento del modelo propuesto, y buscar los hiper-parámetros adecuados para su mejor desempeño.

Para la optimización de parámetros, se usó la técnica de búsqueda arbitraria, o **random search**, la cual consiste en usar un rango amplio de parámetros, escoger los 2 o 3 mejores, y buscar nuevamente parámetros cercanos a ellos. Como la sección 5.2 lo refiere, la segunda versión de la red neuronal, usa una cantidad de parámetros mayor.

Para probar la auto-organización de los agentes, basta ver que la red converge, lo que significa que los agentes de la misma capa, lograr organizarse para dar juntos un estado que le sirva a la capa posterior a tomar buenas decisiones. Sin embargo, otra forma que podemos verlo reflejado es en las diferentes figuras que se irán mostrando por cada problema, las cuales fueron hechas ploteando los estados por cada

agente, y el color representa la acción óptima tomada por dicho agente. Se puede apreciar que toman acciones diferentes unos de otros estando en el mismo estado, y aún así existe una coordinación para poder llegar a la solución correcta.

6.1.1 Bases de datos

Dado que el modelo es discreto, los data sets usados para la validación del modelo son de baja dimensionalidad. En la tabla 6.1, se tiene un resumen de las bases de datos usadas, con la cantidad de atributos y clases por cada una.

Base de datos	Muestras	Clases	Dimensiones
Jain	373	2	2
Flame	240	2	2
Pathbased	300	3	2
Spiral	312	3	2
Compound	399	6	2
Aggregation	788	7	2
R15	600	15	2
Iris	150	3	4
Blood	748	2	3

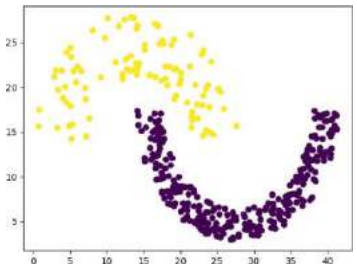
Cuadro 6.1: Descripción de bases de datos usadas

Para poder visualizar mejor las bases de datos de dos dimensiones, se presenta la figura 6.1. Iris, la cual no es de 2 dimensiones, no se puede plotear directamente.

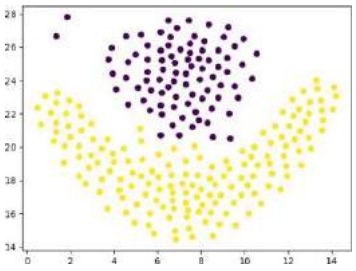
6.1.2 Detalles de implementación

Para estandarizar las pruebas, y que los resultados comparativos sean más objetivos, se usaron los siguientes detalles de implementación:

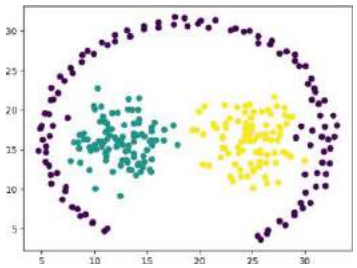
- El número de épocas se fijo en $= 1000$, porque se vió en general que la mayoría de pruebas lograban converger alcanzado ese número de épocas.



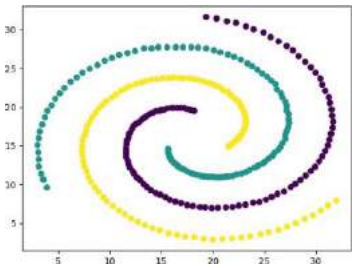
(a) Jain



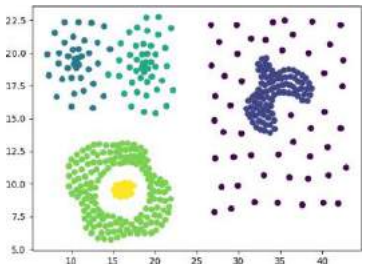
(b) Flame



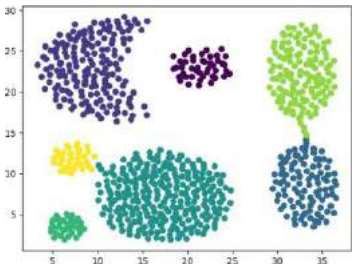
(c) Pathbased



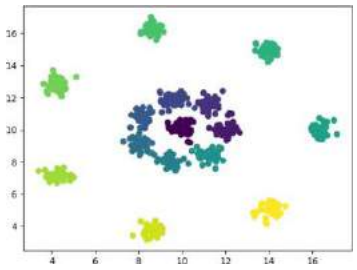
(d) Spiral



(e) Compound



(f) Aggregation



(g) R15

Figura 6.1: Bases de datos de 2 dimensiones usadas

- Las redes neuronales pueden sufrir de *overfitting*, esto significa que solo *memorizan* los datos, y no generalizan. Para diagnosticarlo comúnmente se separa la base de datos en data de entrenamiento y data de prueba, por este motivo se usó a la técnica de validación cruzada k-folds con un $k = 10$.

6.2 VALIDACIÓN DEL MÉTODO PROPUESTO

Para poder validar nuestro modelo propuesto, se realizaron comparaciones con un perceptrón multicapa (MLP). Dado que la data es de baja dimensionalidad, la estructura del MLP es de dos capas, con 10 neuronas cada una. Para el parámetro de factor de aprendizaje, se uso uno del 0.03, lo que nos permite un desempeño de nivel promedio de este modelo.

- Estructura: Dos capas intermedias, con 10 neuronas cada una;
- Factor de aprendizaje ($\eta = 0{,}03$).

En cuanto a la estructura y parámetros usados en la mayoría de las bases de datos de la red IVQ, se usaron los siguientes parámetros

- Coeficiente de aprendizaje, $\alpha = 0{,}05$;
- Coeficiente de comunicación $\beta = 0{,}75$;
- Factor de temperatura $Ft = 0{,}99$;
- Temperatura base $Tb = 0{,}05$;
- Estructura: Una capa intermedia con 2 neuronas en ella;
- Alcance de estados: $r = 2$;
- Sigma $\sigma = 0{,}03$;
- Estrategia de selección de acciones: *Softmax*;
- Número de particiones: $b = 50$.

6.2.1 JAIN

Jain es una de las bases de datos más simples, como lo vemos en la figura 6.1a.

Métrica	MLP	IVQ-Net
Error	5.08772 ±2.49551 %	0.27778 ±0.83333 %
Precision	0.944403 %	0.095 %
Recall	0.92836 %	0.0981481 %
FScore	0.935766 %	0.0965484 %

Cuadro 6.2: Comparación IVQ-Net y MLP en Jain

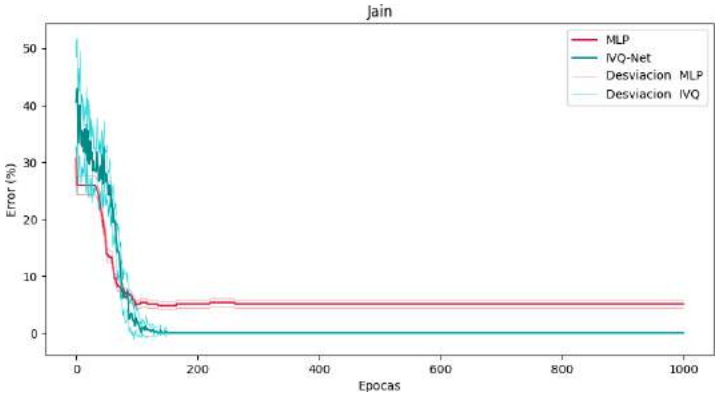


Figura 6.2: Comparación IVQ-Net y MLP en Jain

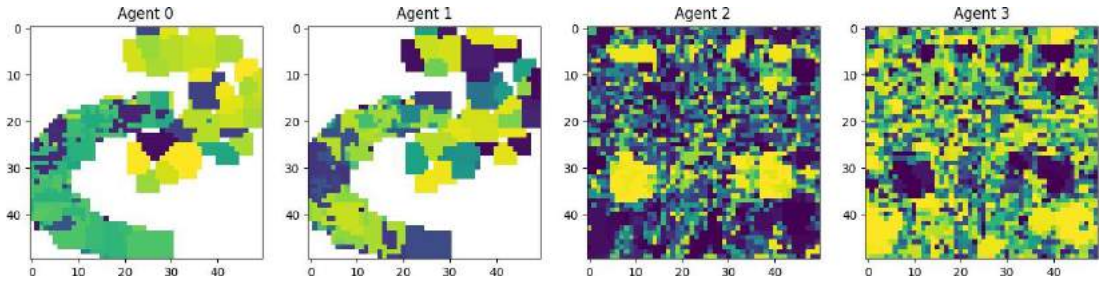


Figura 6.3: Acciones óptimas por estado en Jain

El desempeño de la red IVQ, es muy bueno con 0.27778 % de error, superando el desempeño de el perceptrón multicapa el cual alcanza un 5.08772 %. Como vemos en la figura 6.3 los agentes de la capa oculta, toman diferente decisiones, aunque se trate de la misma clase, dado que son estados diferentes.

6.2.2 FLAME

Aunque en la gráfica 6.4 el desempeño es muy similar, nuestro modelo propuesto se desempeño de mejor manera que, con puntos decimales, y con una mejor desviación estándar como lo vemos en la tabla 6.3.

Métrica	**MLP**	**IVQ-Net**
Error	2.90435 $\pm$5.15912 %	0.869565 $\pm$2.6087 %
Precision	0.971354 %	0.0941176 %
Recall	0.969236 %	0.0875 %
FScore	0.970146 %	0.0906883 %

Cuadro 6.3: Comparación IVQ-Net y MLP en Flame

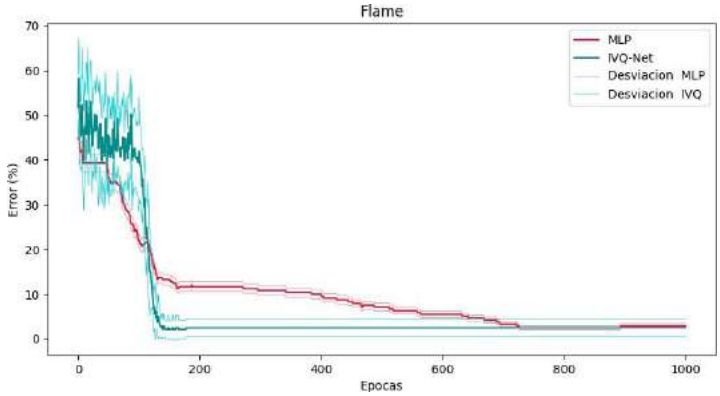


Figura 6.4: Comparación IVQ-Net y MLP en Flame

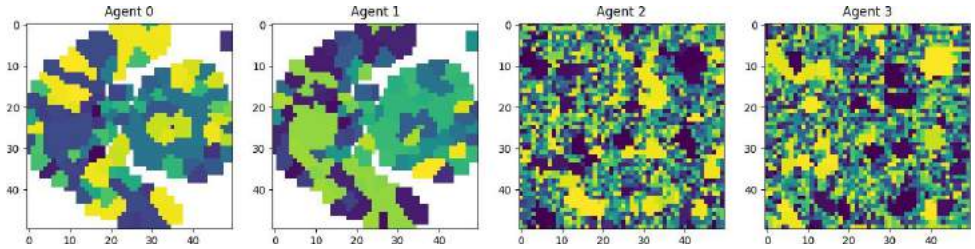


Figura 6.5: Acciones óptimas por estado en Flame

Vemos que el desempeño de IVQ-Learning es superior que el MLP en esta bases de datos. Nuestro modelo converge de manera más rápida, y se estabiliza en un 0.869565 % de error como lo muestra la tabla 6.3 .

6.2.3 PATHBASED

En esta base de datos también se tiene un mejor desempeño en el modelo, que en un perceptrón multicapa como se muestra en al figura 6.6 donde se ve que la red IVQ converge mucho más rápido que el MLP.

Métrica	MLP	IVQ-Net
Error	3.60215 ±3.71684 %	1.96774 ±2.03359 %
Precision	0.969019 %	0.983205 %
Recall	0.965152 %	0.979562 %
FScore	0.967061 %	0.981372 %

Cuadro 6.4: Comparación IVQ-Net y MLP en Pathbased

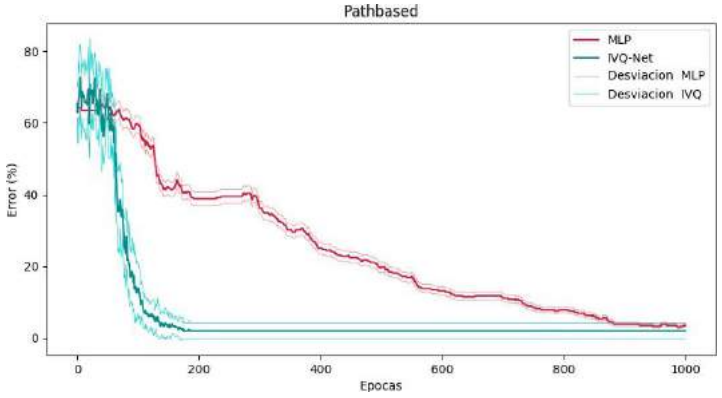


Figura 6.6: Comparación IVQ-Net y MLP en Pathbased

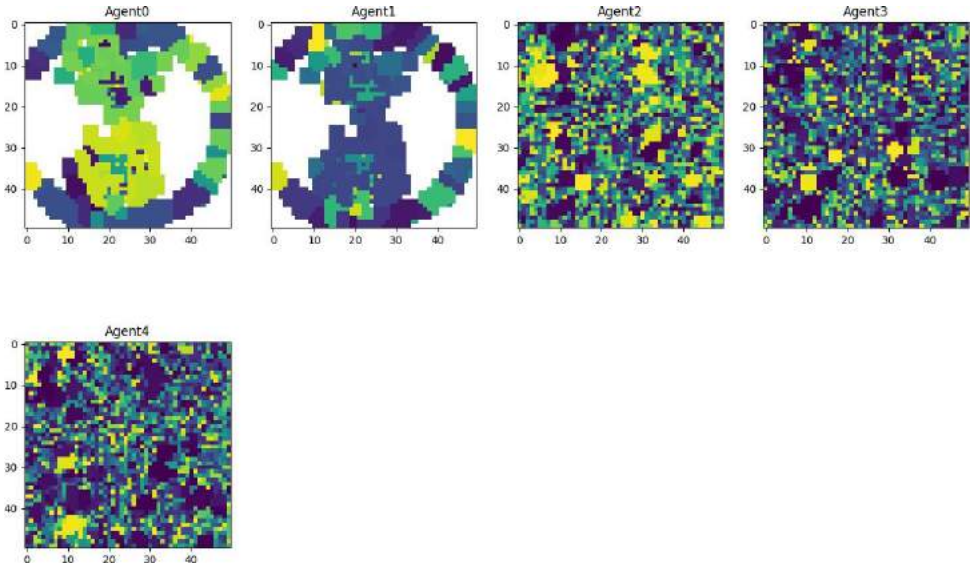


Figura 6.7: Acciones óptimas por estado en Pathbased

Además como la tabla 6.4 muestra, el error final de MLP es 3.60215 % mientras que el modelo propuesto alcanza un 1.96774 % de error, incluso con una desviación estándar menor.

6.2.4 SPIRAL

En esta base de datos es donde se puede notar una gran diferencia de desempeño como lo muestra la tabla 6.5 y la figura 6.8. En este caso, el perceptrón multicapa no converge, mientras que nuestro modelo de red si lo hace.

Métrica	MLP	IVQ-Net
Error	66.0671 ±6.30955 %	0 ±0 %
Precision	0.339697 %	1 %
Recall	nan %	1 %
FScore	nan %	1 %

Cuadro 6.5: Comparación IVQ-Net y MLP en Spiral

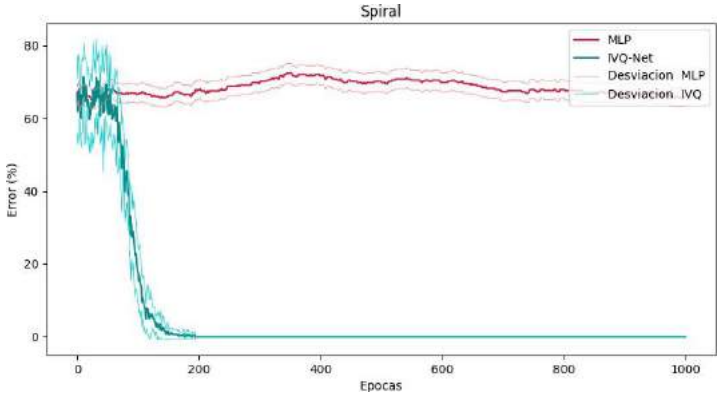


Figura 6.8: Comparación IVQ-Net y MLP en Spiral

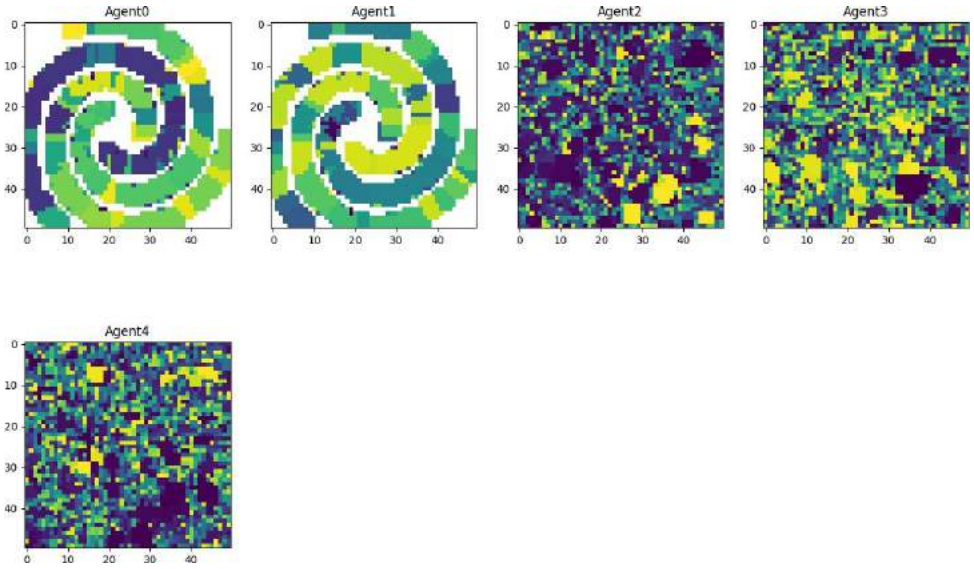


Figura 6.9: Acciones óptimas por estado en Spiral

El motivo que no converge el MLP, es muy probable que sea por la cantidad de capas y neuronas necesarias para poder generar funciones lo suficientemente complejas para que puedan agrupar las clases como se puede ver en la figura 6.1d.

6.2.5 COMPOUND

Como vemos en la tabla 6.6 el modelo propuesto logra un mejor desempeño, obteniendo 3.51873 % mientras que perceptrón multicapa logra solamente un 12.1788 % de error. Igualmente, se tiene una menor desviación en nuestro modelo que en el perceptrón multicapa.

Métrica	MLP	IVQ-Net
Error	12.1788 ±3.09945 %	3.51873 ±1.9871 %
Precision	nan %	0.962121 %
Recall	0.727315 %	0.953981 %
FScore	nan %	0.957916 %

Cuadro 6.6: Comparación IVQ-Net y MLP en Compound

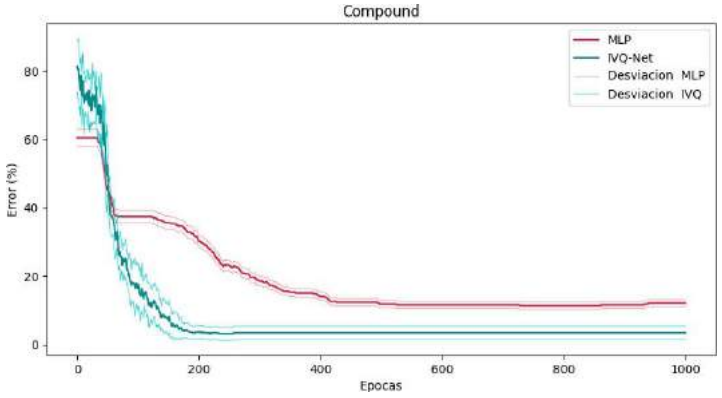


Figura 6.10: Comparación IVQ-Net y MLP en Compound

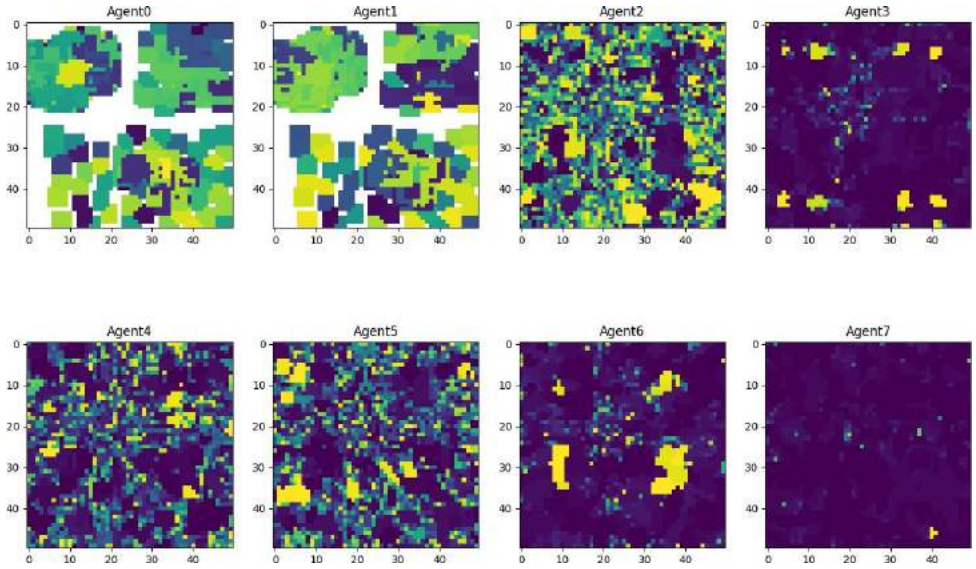


Figura 6.11: Acciones óptimas por estado en Compound

Como la figura 6.10, el la red IVQ converge más rápido que el perceptrón multicapa, de la misma manera que en las anteriores bases de datos.

6.2.6 AGGREGATION

En esta base de datos, tenemos un desempeño perfecto por parte de nuestro modelo, y un desempeño muy cercano en el MLP, como se puede apreciar en la tabla 6.7, es la base de datos donde los modelos más se asemejan en cuanto a desempeño se trata.

Métrica	MLP	IVQ-Net
Error	0.251821 ±0.503954 %	0 ±0 %
Precision	0.995238 %	1 %
Recall	0.998372 %	1 %
FScore	0.996781 %	1 %

Cuadro 6.7: Comparación IVQ-Net y MLP en Aggregation

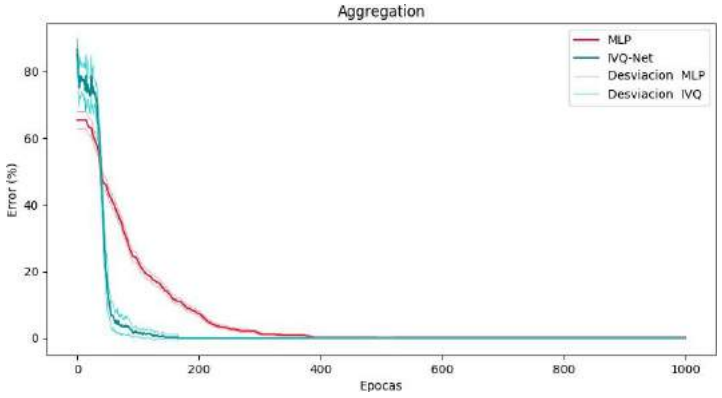


Figura 6.12: Comparación IVQ-Net y MLP en Aggregation

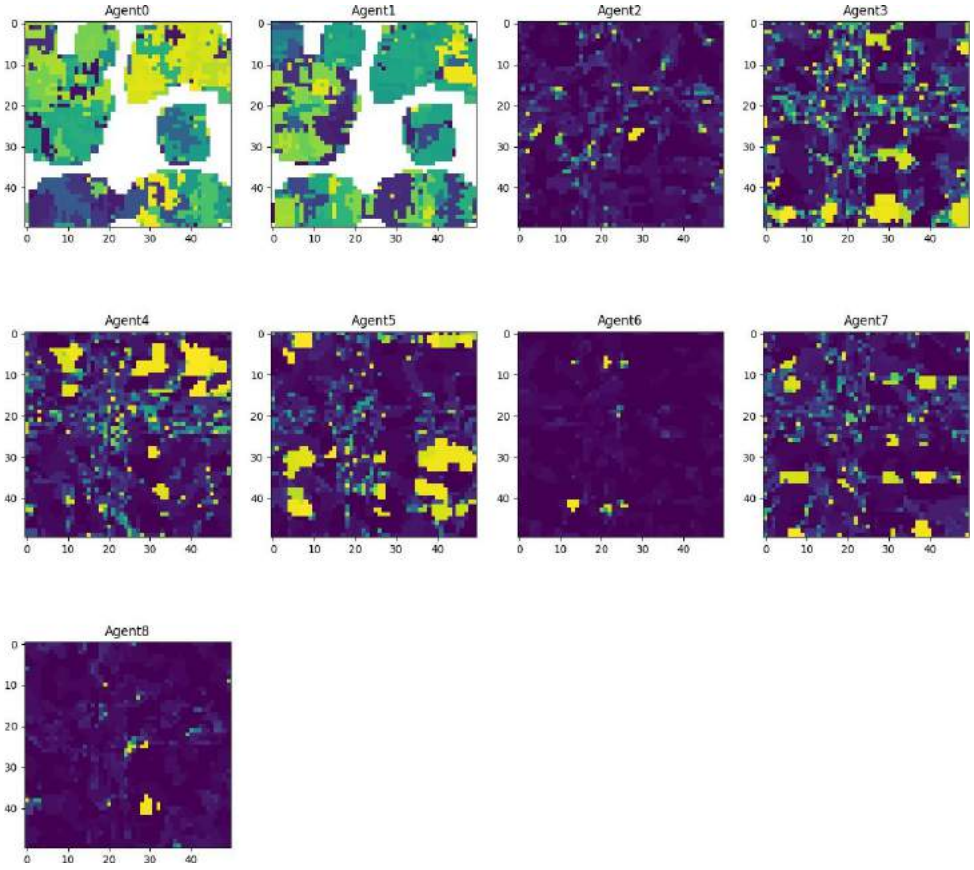


Figura 6.13: Acciones óptimas por estado en Aggregation

Sin embargo, como la figura 6.12 muestra, nuestro modelo logra converger en casi la mitad de épocas que el MLP. Además vemos en la figura 6.13,

6.2.7 R15

En este caso, dado que la cantidad de clases es mayor que en anteriores ejemplos al igual que el número de muestras, se seteó el número de divisiones a 30. A pesar de este ajuste necesario, vemos en la tabla 6.8 que el perceptrón multicapa superó

el rendimiento en el MLP con 1.83333 % de error versus el 3.51873 % de error de nuestro modelo.

Métrica	MLP	IVQ-Net
Error	1.83333 $\pm$1.89297 %	3.51873 $\pm$1.9871 %
Precision	0.985333 %	0.962121 %
Recall	0.981667 %	0.953981 %
FScore	0.983493 %	0.957916 %

Cuadro 6.8: Comparación IVQ-Net y MLP en R15

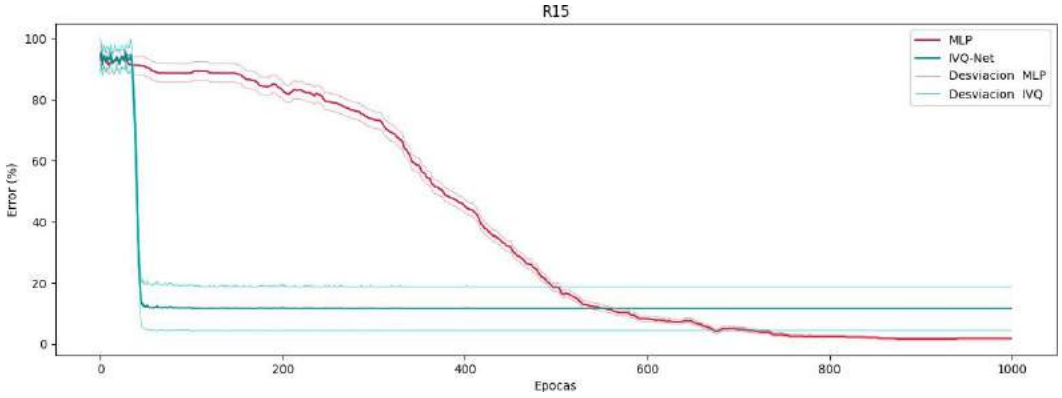


Figura 6.14: Comparación IVQ-Net y MLP en R15

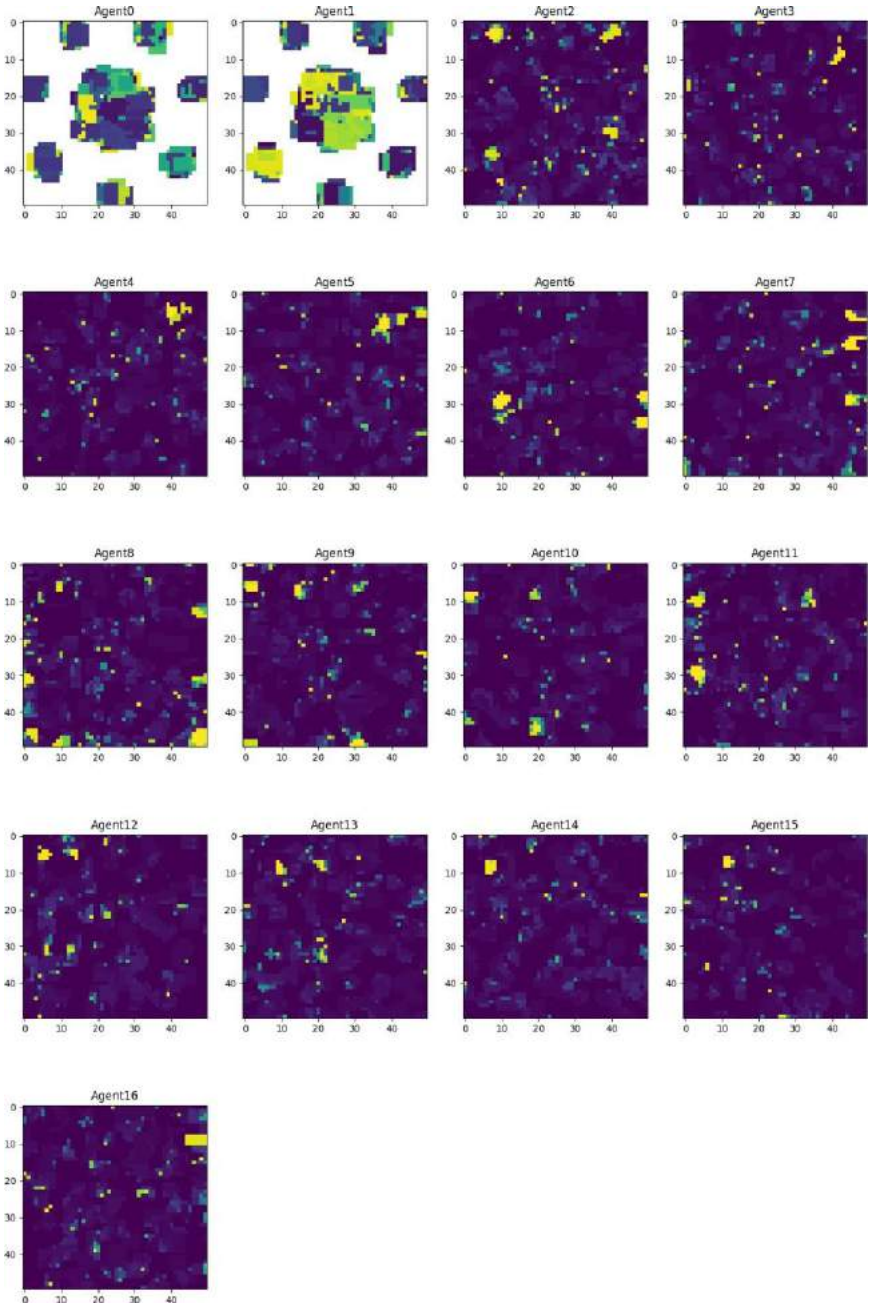


Figura 6.15: Acciones óptimas por estado en R15

En la figura 6.14, se puede apreciar que nuestro modelo converge mucho mas rápido que el MLP, sin embargo, el MLP llega a tener un mejor desempeño al final de las 1000 épocas. Otra cosa a notar es la gran desviación de nuestro modelo desde el momento que converge.

6.2.8 IRIS

En esta base de datos vemos de igual manera que el perceptrón multicapa superó el rendimiento de nuestro modelo propuesto, como se muestra en la tabla 6.9 donde el MLP alcanza un 4 % de error mientras que el modelo propuesto alcanza el 5.91667 %.

Métrica	MLP	IVQ-Net
Error	4 $\pm$5.33333 %	5.91667 $\pm$6.87639 %
Precision	0.966032 %	0.946587 %
Recall	0.96 %	0.941111 %
FScore	0.962968 %	0.943801 %

Cuadro 6.9: Comparación IVQ-Net y MLP en Iris

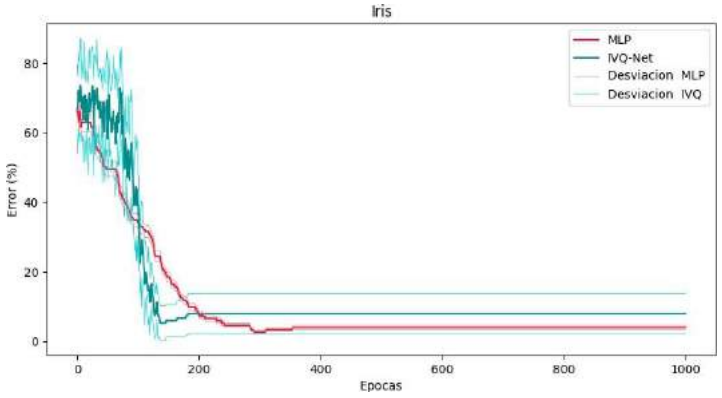


Figura 6.16: Comparación IVQ-Net y MLP en Iris

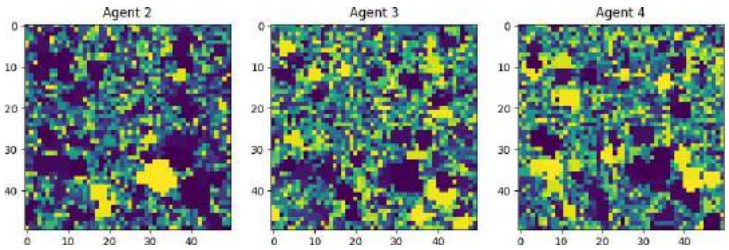


Figura 6.17: Acciones óptimas por estado en Iris

Como ese puede apreciar en al figura 6.16, si bien nuestro modelo converge más rápido que el MLP, la diferencia no es tanta como en las otras bases de datos, a pesar de esto, el MLP logra una mejor convergencia que nuestro modelo. Hay que notar que en la figura 6.17 no se grafican los agentes de la capa intermedia, dado que los estados ya no son dos dimensiones como en las anteriores bases de datos, sino que son de cuatro dimensiones.

6.2.9 Blood

Por último, esta base de datos es la más compleja, dado que se presenta data no balanceada. Vemos en la tabla 6.10 que el desempeño de los dos modelos llegan a 20.9892 % de error en el MLP y a 29.6775 %, el peor desempeño de todas las bases de datos.

Métrica	MLP	IVQ-Net
Error	20.9892 $\pm$2.99163 %	29.6775 $\pm$3.55115 %
Precision	0.7248 %	0.583295 %
Recall	0.607482 %	0.577236 %
FScore	0.659761 %	0.580141 %

Cuadro 6.10: Comparación IVQ-Net y MLP en Blood

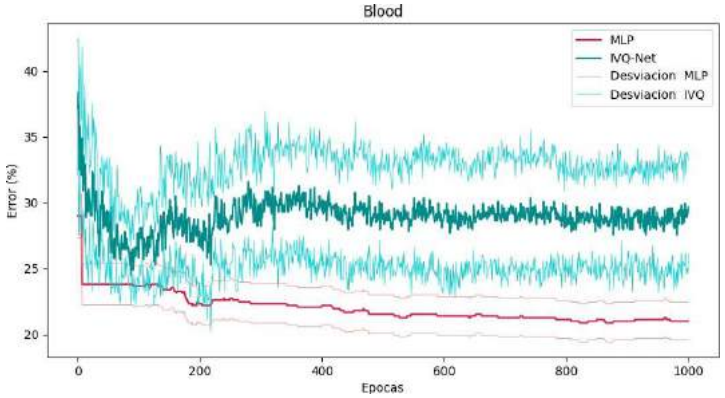


Figura 6.18: Comparación IVQ-Net y MLP en Blood

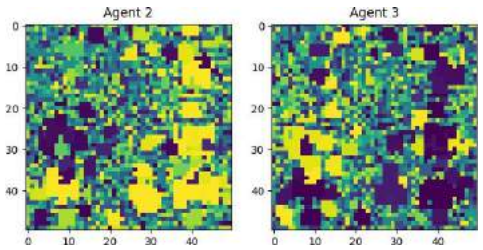


Figura 6.19: Acciones óptimas por estado en Blood

Vemos en la figura 6.18 que la desviación del MLP es alta durante todo el entrenamiento. Sin embargo, vemos que la desviación de nuestro modelo, es la más grande de todos las bases de datos, donde incluso se puede ver que no hay una convergencia, sino que hasta el final de las 1000 épocas existen picos muy notorios. Al igual que Iris, no se grafican los agentes de la capa oculta en la figura 6.19, porque los estados de estos no son de dos dimensiones sino de tres.

6.2.10 Discusión

Se puede apreciar, que con distintas bases de datos, el modelo puede generalizar, independientemente del número de clases o dimensiones que pueda tener la base de datos. En algunos casos, el modelo supera al desempeño del perceptrón multicapa, mientras que en otros casos es al contrario. Es posible que esto se deba a la

naturaleza de la base de datos, dado que los agentes de nuestro modelo, discretizan el espacio de varias dimensiones, lo que se podría asemejar a knn o incluso las redes growing neural gas resuelven los problemas a nivel espacial.

Un resumen del desempeño de estas redes, se aprecia en la tabla 6.11.

Base de datos	MLP	IVQ-Net
Jain	5.08772 ±2,49551 %	**0 ±0 %**
Flame	2.90435 ±5,15912 %	**2.48623 ±2,03359 %**
Pathbased	3.60215 ±3,71684 %	**1.96774 ±2,03359 %**
Spiral	66.0671 ±6,30955 %	**0 ±0 %**
Compound	12.1788 ±3,09945 %	**3.51873 ±1,9871 %**
Aggregation	0.251821 ±0,503954 %	**0 ±0 %**
R15	**1.83333 ±1,89297 %**	3.51873 ±1,9871 %
Iris	**4 ±5,33333** %	5.91667 ±6,87639 %
Blood	**20.9892 ±2.99163** %	29.6775 ±3.55115 %

Cuadro 6.11: Resumen de la precisión en las bases de datos

6.3 Análisis de parámetros

Una de las características de esta red, es la cantidad de parámetros usados. Como lo dijimos anteriormente, en las pruebas se uso la búsqueda random de parámetros para poder encontrar la combinación más optima. Sin embargo, es necesaria hacer un análisis más profundo de ellos, comparando el desempeño con diferentes valores de estos. Para esto, se hizo experimentos con la base de datos iris, que es posible que sea la más conocida y usada para testear modelos de aprendizaje supervisados con baja dimensionalidad.

6.3.1 Alpha (α)

El coeficiente de aprendizaje α, se podría interpretar como el valor de aprendizaje de los agentes con respecto de la función de valor-acción pasada.

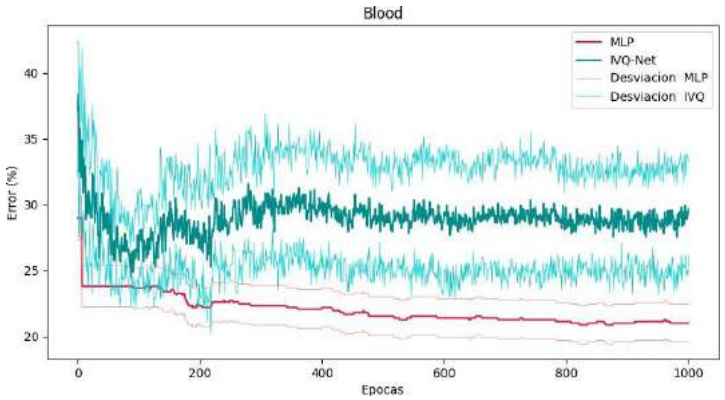


Figura 6.20: Comparación de desempeño según alpha

Como se puede observar en la imagen 6.20, mientras los agentes aprendan más rápido, es decir, el valor alpha sea mayor, el algoritmo convergerá más rápido, sin embargo, existe un umbral el cual, si es que se excede, el modelo no converge. Además como se puede apreciar, en el momento en que converge, los valores altos de α, tienen un valor de error más alto.

6.3.2 Beta (β)

Al variar el coeficiente de influencia, estamos variando cuánto de lo aprendido por la capa l, se comunicará a la capa $l-1$. Sin embargo, como podemos ver en la figura 6.21, este parámetro es bastante flexible en cuanto a convergencia.

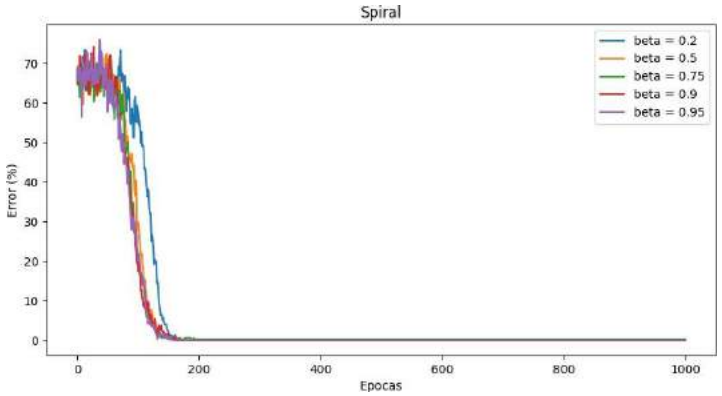


Figura 6.21: Comparación de desempeño según beta

6.3.3 Factor de Temperatura

Aunque es un coeficiente independiente del modelo, es un factor importante para los resultados presentados en este trabajo. Al modificar este factor, estamos modificando la velocidad del descenso de la temperatura, lo cual se puede apreciar en la figura 6.22.

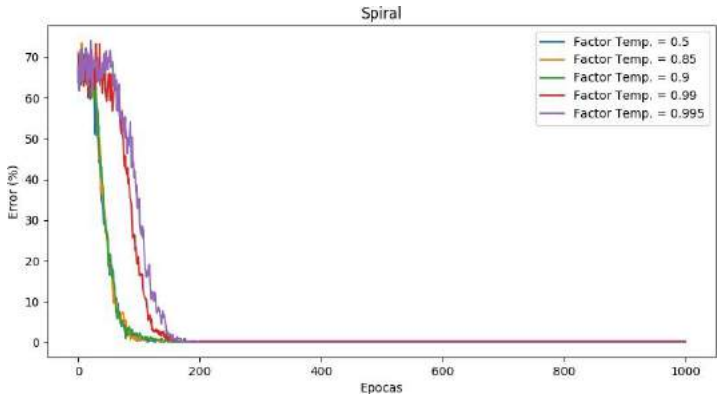


Figura 6.22: Comparación de desempeño según el factor de temperatura

6.3.4 Número de divisiones Propagación de estados

Estos parámetros están directamente relacionados. Dependiendo de la base da datos usada, esto parámetros deberá ser modificado. Si es que la base de datos es muy compleja, es mejor que se aumente el número de divisiones, mientras que para bases de datos simples, puede quedarse bajo. Además este parámetro, es muy influyente en cuanto la cantidad de RAM usada por el computador, dado que si es que se tienen muchas dimensiones, incrementar mucho el número de divisiones generará una cantidad muy grande estados, que es posible que computadores simples no puedan manejarlo.

Es importante notar en las figuras 6.23, 6.24, y 6.25 que si es que se escoge un número de divisiones muy bajo (5 divisiones en esta base de datos), el algoritmo llega a su punto de convergencia rápidamente, pero el desempeño es bajo, esto sucede aunque el radio de expansión $R = 2$. Por otro lado, vemos que el algoritmo se estabiliza cuando el radio de expansión es $R = 2$, aún con divisiones altas de 90, sin embargo, vemos ue no llega a su mejor desempeño, aunque con divisiones de 15 hasta 30, llega a un desempeño perfecto. Cuando se tiene divisiones bajas (15),

basta con un radio bajo como 1, sin embargo al aumentar las divisiones, el radio de expansión, también debe de aumentar.

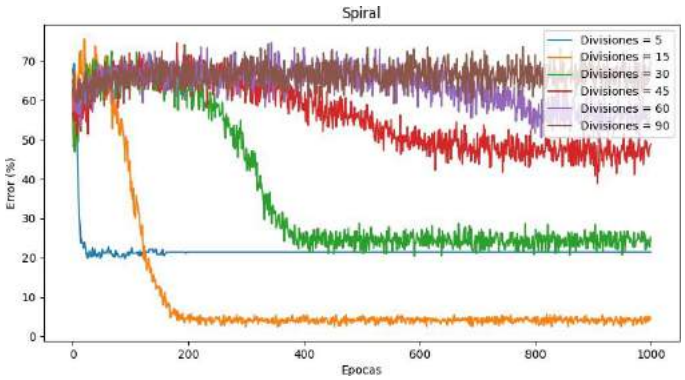


Figura 6.23: Desempeño según número de divisiones, con R = 0

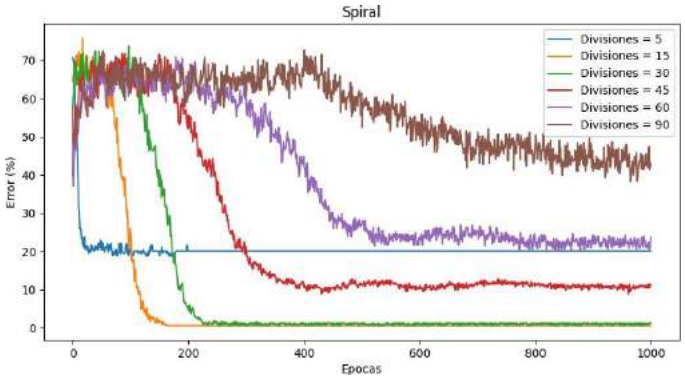


Figura 6.24: Desempeño según número de divisiones, con R = 1

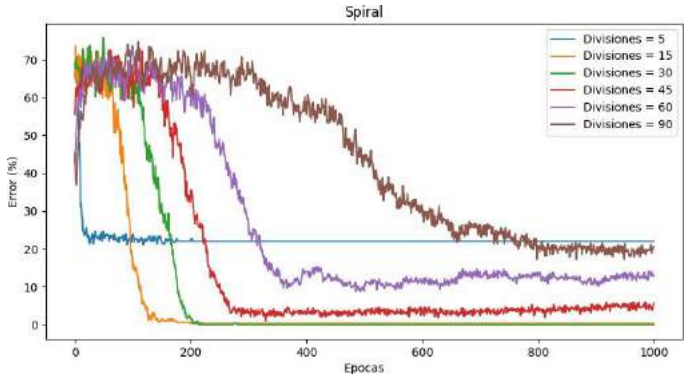


Figura 6.25: Desempeño según número de divisiones, con R = 2

6.3.5 Estructura

Este parámetro es influyente para la rapidez de este modelo como se ve en la figura 6.26 y 6.27, sobre todo el número de neuronas por capa. Primero, se analizará aumentando ligeramente el número de neuronas en la capa intermedia. Este afecta directamente al tiempo, esto es gracias que la expansión del conocimiento en los estados, aumenta exponencialmente con respecto a las dimensiones; y dado que para la última capa del modelo, el número de dimensiones es igual al número de neuronas en la capa anterior, se complica el entrenamiento.

En cuanto al análisis aumentando el número de capas, se puede ver que al aumentar el número de capas, si bien aumenta el tiempo de convergencia y el error; los agentes siguen aprendiendo. Esto significa que tanto el algoritmo de aprendizaje como el modelo es robusto, soportando múltiples capas.

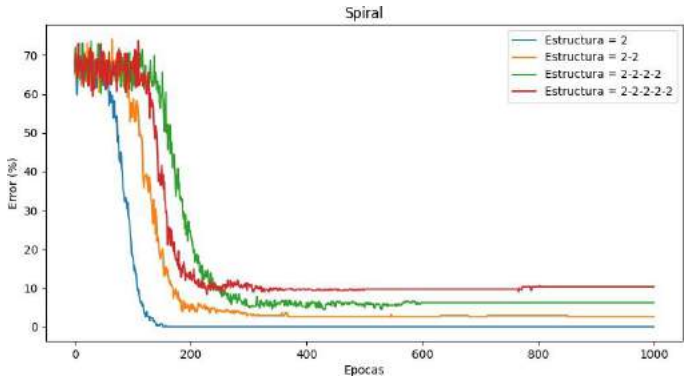


Figura 6.26: Desempeño según la estructura de la red con multiples capas

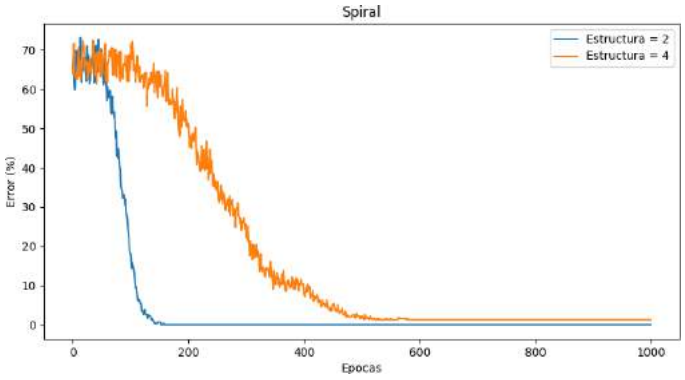


Figura 6.27: Desempeño según la estructura de la red con multiples neuronas

7 CONCLUSIONES Y TRABAJOS FUTUROS

7.1 CONCLUSIONES

Se propuso un nuevo paradigma de aprendizaje para redes neuronales de aprendizaje supervisado, que está basado en la influencia entre neuronas, siendo diferente al tradicional modelo basado en la propagación del error por el método del gradiente. Este nuevo paradigma representa una mayor cercanía a los sistemas naturales en donde se sabe la influencia de los diferentes elementos entre ellos pero no se modela la forma en que se realiza.

Además, se propuso una forma de discretización de estados distinta, que inspirado en la función de vecindad de los mapas auto-organizativos (SOM), es de gran utilidad para nuestra red, dado que los estados que no se lograban visitar en la fase de entrenamiento con la primera versión del algoritmo, terminan alcanzando el conocimiento gracias a la expansión del conocimiento de sus estados vecinos cercanos.

Otro aspecto bastante resaltante a notar, es el algoritmo de aprendizaje. Si bien, no nos hemos enfocado en este trabajo en analizar el algoritmo *backpropagation* contra *valor de influencia*, es de mucho valor que este modelo no use el algoritmo por excelencia de muchos otros modelos neuronales, y que tenga un desempeño similar. Esto se ve probado especialmente en el desempeño de la red con diferentes estructuras, cuando al aumentar el número de neuronas, e incluso el número de capas, el modelo es capaz de converger de manera óptima.

Para finalizar, se ha probado con resultados satisfactorios que el modelo propuesto funciona en la solución de problemas de aprendizaje supervisado al tener resultados comparables con el perceptrón con múltiples capas.

7.2 LIMITACIONES

La principal limitación del modelo neuronal presentado, es que actualmente funciona en bases de datos con baja dimensionalidad, dado que el número de estados crece exponencialmente con respecto del número de dimensiones de la data, por el mismo motivo, el número de neuronas en las capas intermedias también deben de ser pocas. Esta limitación es causada porque los agentes neuronales trabajan de forma discreta, no permitiendo usar la red en problemas más complejos.

7.3 TRABAJOS FUTUROS

Este modelo, al ser pionero, y usar neuronas como agentes básicos para su desempeño, tiene muchas vías de mejoras las cuales se describirán a continuación:

- Como primer paso, es importante crear el algoritmo IVQ-Learning para agentes continuos, para de esta manera implementar esta nueva versión en la red neuronal, así dejamos de discretizar, y trabajamos directamente en un campo continuo.

- Además, dado que una característica esencial del aprendizaje por refuerzo, los agentes tienen en cuenta la variable tiempo, es decir, están diseñados para desempeñarse en juegos que no son repetitivos. Por lo tanto si es que combinamos esto, y usamos una red neuronal recurrente como interpolador de funciones para el agente continuo, podría funcionar esta red muy bien para series de tiempo.

- Por último, la auto-organización, es un tema esencial en el cerebro biológico, y un problema aún latente en las redes neuronales artificiales actuales es el olvido catastrófico. Estas redes aprenden una tarea, pero al intentar aprender otra, olvidan la primera. Haciendo más pruebas, donde pueden estar involucrados pesos sinápticos, consolidación de memoria, etc. este modelo debería ser capaz de enfrentar el olvido catastrófico de una mejor manera.

7.4 PRINCIPALES CONTRIBUCIONES

- Nuevo modelo para aprendizaje supervisado basado en aprendizaje por refuerzo con valores de influencia.

- Algoritmo de aprendizaje de valor de influencia.
- Algoritmo para discretización de estados para agentes de aprendizaje por refuerzo.
- Publicación de articulo científico [53]: *André Valdivia, Jose Herrera Quispe, and Dennis Barrios-Aranibar. 2018. A new approach for supervised learning based influence value reinforcement learning. In Proceedings of the 2nd International Conference on Machine Learning and Soft Computing (ICMLSC '18). ACM, New York, NY, USA, 24-28. DOI: https://doi.org/10.1145/3184066.3184094.*

BIBLIOGRAFÍA

1. O. Amosov, Y. Ivanov y S. Zhiganov. "Human Localization in the Video Stream Using the Algorithm Based on Growing Neural Gas and Fuzzy Inference". *Procedia Computer Science* 103, 2017, págs. 403-409.
2. D. Barrios-Aranibar y L. M. G. Goncalves. "Learning Coordination in Multi-Agent Systems Using Influence Value Reinforcement Learning". En: *Seventh International Conference on Intelligent Systems Design and Applications (ISDA 2007)*. 2007, págs. 471-478.
3. R. Bellman. "Dynamic programming and Lagrange multipliers". *Proceedings of the National Academy of Sciences* 42:10, 1956, págs. 767-769.
4. G. A. Carpenter y S. Grossberg. "ART 2: Self-organization of stable category recognition codes for analog input patterns". *Applied optics* 26:23, 1987, págs. 4919-4930.
5. G. A. Carpenter y S. Grossberg. "ART 3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures". *Neural networks* 3:2, 1990, págs. 129-152.
6. G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds y D. B. Rosen. "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps". *IEEE Transactions on neural networks* 3:5, 1992, págs. 698-713.
7. G. A. Carpenter, S. Grossberg y J. H. Reynolds. "ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network". *Neural networks* 4:5, 1991, págs. 565-588.
8. G. A. Carpenter, S. Grossberg y D. B. Rosen. "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system". *Neural networks* 4:6, 1991, págs. 759-771.
9. J.-H. Chen, M.-C. Su, R. Cao, S.-C. Hsu y J.-C. Lu. "A self organizing map optimization based image recognition and processing model for bridge crack inspection". *Automation in Construction* 73, 2017, págs. 58-66.

10. C. Cortes, X. Gonzalvo, V. Kuznetsov, M. Mohri y S. Yang. "AdaNet: Adaptive Structural Learning of Artificial Neural Networks". *arXiv preprint arXiv:1607.01097*, 2016.

11. G. Cybenko. "Approximation by superpositions of a sigmoidal function". *Mathematics of Control, Signals, and Systems (MCSS)* 2:4, 1989, págs. 303-314.

12. L. N. De Castro. *Fundamentals of natural computing: basic concepts, algorithms, and applications*. CRC Press, 2006.

13. L. Deng, J. Li, J.-T. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams y col. "Recent advances in deep learning for speech research at Microsoft". En: *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE. 2013, págs. 8604-8608.

14. P. J. A. Dennis Barrios-Aranibar. "Learning strategies for coordination of multirobot systems: a robot soccer application". *XVI Congreso Brasileiro de Automática*, 2006.

15. Y Du, C Yuan, W Hu y S. Maybank. "Spatio-temporal self-organizing map deep network for dynamic object detection from videos". En: *IEEE Conference on Computer Vison and Pattern Recognition 2017*. IEEE Computer Society. 2017.

16. Y. Dupont, M. Dinarelli e I. Tellier. "Label-Dependencies Aware Recurrent Neural Networks". *arXiv preprint arXiv:1706.01740*, 2017.

17. S. Elfwing, E. Uchibe y K. Doya. "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning". *arXiv preprint arXiv:1702.03118*, 2017.

18. H. J. Fariman, S. A. Ahmad, M. H. Marhaban, M. A. Ghasab y P. H. Chappell. "Hand movements classification for myoelectric control system using adaptive resonance theory". *Australasian Physical & Engineering Sciences in Medicine* 39:1, 2016, págs. 85-102.

19. C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel y D. Wierstra. "Pathnet: Evolution channels gradient descent in super neural networks". *arXiv preprint arXiv:1701.08734*, 2017.

20. S. Ghosh-Dastidar y H. Adeli. "Spiking neural networks". *International journal of neural systems* 19:04, 2009, págs. 295-308.

21. S. Grossberg. "Competitive learning: From interactive activation to adaptive resonance". *Cognitive science* 11:1, 1987, págs. 23-63.

22. S. Grossberg. "Competitive learning: From interactive activation to adaptive resonance". *Cognitive science* 11:1, 1987, págs. 23-63.

23. H. Guan, X. Xue y A. Zhiyong. "Online video tracking using collaborative convolutional networks". En: *Multimedia and Expo (ICME), 2016 IEEE International Conference on*. IEEE. 2016, págs. 1-6.

24. A. Hassan. "SENTIMENT ANALYSIS WITH RECURRENT NEURAL NETWORK AND UNSUPERVISED NEURAL LANGUAGE MODEL", 2017.

25. S. Haykin. *Neural networks and learning machines*. Vol. 3. Pearson Upper Saddle River, NJ, USA: 2009.

26. K. He, X. Zhang, S. Ren y J. Sun. "Deep residual learning for image recognition". En: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, págs. 770-778.

27. G. E. Hinton. "Distributed representations", 1984.

28. A. L. Hodgkin y A. F. Huxley. "A quantitative description of membrane current and its application to conduction and excitation in nerve". *The Journal of physiology* 117:4, 1952, pág. 500.

29. J. J. Hopfield. "Neural networks and physical systems with emergent collective computational abilities". *Proceedings of the national academy of sciences* 79:8, 1982, págs. 2554-2558.

30. J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska y col. "Overcoming catastrophic forgetting in neural networks". *Proceedings of the National Academy of Sciences*, 2017, pág. 201611835.

31. T. Kohonen. "Self-organized formation of topologically correct feature maps". *Biological Cybernetics* 43:1, 1982, págs. 59-69.

32. M. Kowalski, J. Naruniec y T. Trzcinski. "Deep Alignment Network: A convolutional neural network for robust face alignment". *arXiv preprint arXiv:1706.01789*, 2017.

33. D. I. Kumar y M. R. Kounte. "Comparative study of self-organizing map and deep self-organizing map using MATLAB". En: *Communication and Signal Processing (ICCSP), 2016 International Conference on*. IEEE. 2016, págs. 1020-1023.

34. Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard y L. D. Jackel. "Backpropagation applied to handwritten zip code recogni- tion". *Neural computation* 1:4, 1989, págs. 541-551.

35. S. Loiselle, J. Rouat, D. Pressnitzer y S. Thorpe. "Exploration of rank order coding with spiking neural networks for speech recognition". En: *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*. Vol. 4. IEEE. 2005, págs. 2076-2080.

36. W. Maass. "Networks of spiking neurons: the third generation of neural network models". *Neural networks* 10:9, 1997, págs. 1659-1671.

37. W. S. McCulloch y W. Pitts. "A logical calculus of the ideas immanent in nervous activity". *The bulletin of mathematical biophysics* 5:4, 1943, págs. 115-133.

38. V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver y K. Kavukcuoglu. "Asynchronous methods for deep reinforcement learning". En: *International Conference on Machine Learning*. 2016, págs. 1928-1937.

39. V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra y M. Riedmiller. "Playing atari with deep reinforcement learning". *arXiv preprint arXiv:1312.5602*, 2013.

40. N. Morgan y H. Bourlard. "Continuous speech recognition using multilayer perceptrons with hidden Markov models". En: *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*. IEEE. 1990, págs. 413-416.

41. D. Nguyen-Tuong y J. Peters. "Model learning for robot control: a survey". *Cognitive processing* 12:4, 2011, págs. 319-340.

42. P. Panda, J. M. Allred, S. Ramanathan y K. Roy. "ASP: Learning to Forget with Adaptive Synaptic Plasticity in Spiking Neural Networks". *arXiv preprint arXiv:1703.07655*, 2017.

43. F. Piekniewski, E. Izhikevich, B. Szatmary y C. Petre. *Spiking neural network object recognition apparatus and methods*. US Patent App. 13/465,918. 2012.

44. A. M. Rather, A. Agarwal y V. Sastry. "Recurrent neural network and a hybrid model for prediction of stock returns". *Expert Systems with Applications* 42:6, 2015, págs. 3234-3241.

45. D. Reid, A. J. Hussain y H. Tawfik. "Financial time series prediction using spiking neural networks". *PloS one* 9:8, 2014, e103656.

46. C. Robert. *Machine learning, a probabilistic perspective*. 2014.

47. I. Rocco, R. Arandjelović y J. Sivic. "Convolutional neural network architecture for geometric matching". *arXiv preprint arXiv:1703.05593*, 2017.

48. D. E. Rumelhart, G. E. Hinton y R. J. Williams. *Learning internal representations by error propagation*. Inf. téc. DTIC Document, 1985.

49. H. Sak, A. Senior y F. Beaufays. "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition". *arXiv preprint arXiv:1402.1128*, 2014.

50. W. S. Sarle. "Why statisticians should not FART". *Cary, NC, USA*, 1995.

51. A. A. Sherstov y P. Stone. "Function approximation via tile coding: Automating parameter choice". En: *International Symposium on Abstraction, Reformulation, and Approximation*. Springer. 2005, págs. 194-205.

52. B. Shi, X. Bai y C. Yao. "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.

53. A. Valdivia, J. H. Quispe y D. Barrios-Aranibar. "A New Approach for Supervised Learning Based Influence Value Reinforcement Learning". En: *Proceedings of the 2Nd International Conference on Machine Learning and Soft Computing*. ICMLSC '18. ACM, Phu Quoc Island, Viet Nam, 2018, págs. 24-28. ISBN: 978-1-4503-6336-5. DOI: 10.1145/3184066.3184094. URL: http://doi.acm.org/10.1145/3184066.3184094.

54. H. Van Hasselt. "Reinforcement learning in continuous state and action spaces". En: *Reinforcement learning*. Springer, 2012, págs. 207-251.

55. R. Venkateswarlu y R. V. Kumari. "Novel approach for speech recognition by using self—Organized maps". En: *Emerging Trends in Networks and Computer Communications (ETNCC), 2011 International Conference on*. IEEE. 2011, págs. 215-222.

56. J. Weston, S. Chopra y A. Bordes. "Memory Networks". *ArXiv e-prints*, 2014. arXiv: 1410.3916 [cs.AI].

57. J. R. Williamson. "Gaussian ARTMAP: A neural network for fast incremental learning of noisy multidimensional maps". *Neural networks* 9:5, 1996, págs. 881-897.

58. W. Xie, J. A. Noble y A. Zisserman. "Microscopy cell counting with fully convolutional regression networks". En: *MICCAI 1st Workshop on Deep Learning in Medical Image Analysis*. 2015.

59. Z. Zhang, Q. Wu, Z. Zhuo, X. Wang y L. Huang. "Wavelet transform and texture recognition based on spiking neural network for visual images". *Neurocomputing* 151, 2015, págs. 985-995.

60. S. Zweig y L. Wolf. "InterpoNet, A brain inspired neural network for optical flow dense interpolation". *arXiv preprint arXiv:1611.09803*, 2016.