Propiedades computacionales del umbral neuronal y el código *Sparse* en el proceso de discriminación de odorantes

Jessica López-Hazas Sacristán

Máster en Investigación
e Innovación en TIC

MÁSTERES
DE LA UAM
2017 - 2018

Escuela Politécnica Superior

UAM Universidad Autónoma de Madrid

Universidad Autónoma de Madrid
Escuela politécnica superior

Trabajo Fin de Máster

PROPIEDADES COMPUTACIONALES DEL UMBRAL NEURONAL Y EL CÓDIGO *SPARSE* EN EL PROCESO DE DISCRIMINACIÓN DE ODORANTES

Máster Universitario en Investigación e Innovación en TIC

Autora: LÓPEZ-HAZAS SACRISTÁN, Jessica
Tutores: RODRÍGUEZ ORTIZ, Francisco de Borja
MONTERO MONTERO, Aarón
Departamento de Ingeniería Informática

Septiembre 2018

PROPIEDADES COMPUTACIONALES DEL UMBRAL NEURONAL Y EL CÓDIGO *SPARSE* EN EL PROCESO DE DISCRIMINACIÓN DE ODORANTES

Autora: LÓPEZ-HAZAS SACRISTÁN, Jessica
Tutores: RODRÍGUEZ ORTIZ, Francisco de Borja
MONTERO MONTERO, Aarón

Grupo de Neurocomputación Biológica (GNB)
Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Septiembre 2018

Resumen

Comprender los mecanismos que controlan la codificación neuronal es fundamental para entender la forma en que se procesa la información en las redes neuronales. Uno de estos mecanismos es el umbral de disparo neuronal. Existen diversas observaciones experimentales e hipótesis sobre las posibles propiedades computacionales que podrían controlar las dinámicas del umbral de disparo, entre las que destacan su papel a la hora de asegurar que el código neuronal sea óptimo y su influencia en el proceso de discriminación de estímulos en sistemas sensoriales.

Algunas investigaciones llevadas a cabo en el Grupo de Neurocomputación Biológica de la UAM sobre el sistema olfativo de los insectos apuntan a que una distribución heterogénea en los umbrales de las células de Kenyon en el cuerpo fungiforme es uno de los factores que regula la excitabilidad de las neuronas y posibilita la aparición del código disperso empleado por este sistema para representar la información y mejorar su capacidad de discriminación de odorantes.

Dentro de esta línea de investigación, el objetivo que se ha propuesto en este TFM es el desarrollo de un modelo del sistema olfativo del insecto que permita explorar la influencia de la distribución de umbrales neuronales en su capacidad de discriminación, la generación del código disperso y su eficiencia energética.

Para alcanzar este objetivo, se ha desarrollado un nuevo modelo del sistema olfativo de la langosta basado en redes neuronales y aprendizaje supervisado que incluye un algoritmo de aprendizaje capaz de ajustar los umbrales de disparo de las neuronas KCs. Además, el algoritmo también cuenta con un término de regulación de la actividad de las KCs que permite controlar su nivel de activación.

Para la validación del funcionamiento del modelo propuesto se han utilizado tres conjuntos de datos con patrones de diversos niveles de complejidad. La conclusión principal que se ha obtenido es que el algoritmo de aprendizaje desarrollado es capaz de regular la actividad de las neuronas KCs a partir del ajuste de sus umbrales de disparo y obtiene mejores resultados cuando converge a un estado similar al encontrado en la biología. Además, también es capaz de ajustar las características de la distribución de umbrales de disparo en las KCs para adaptarla a la complejidad de los patrones que se están clasificando. Para ello, estable diferentes balances entre la cantidad de neuronas que presentan un umbral alto con comportamiento especialista y un umbral bajo con comportamiento generalista, lo que es coherente con otros estudios realizados en el GNB sobre la influencia balance de neuronas especialistas y generalistas en el reconocimiento de patrones. Aparte de esto, el modelo es capaz de conseguir resultados iguales o mejores a los obtenidos utilizando SVMs para la clasificación de odorantes captados por narices electrónicas. Por tanto, los resultados obtenidos muestran que el umbral neuronal tiene una gran importancia a la hora de determinar la capacidad de discriminación del sistema y controlar la generación del código disperso para la representación de la información.

Palabras Clave

Neurociencia computacional, sistema olfativo, umbral neuronal, células de Kenyon, redes neuronales bio-inspiradas, aprendizaje automático, modelado de sistemas biológicos, reconocimiento de patrones, heterogeneidad neuronal, código disperso, eficiencia energética

Abstract

Understanding the mechanisms that control neural coding is fundamental to comprehend the way in which information is processed in neural networks. One of these mechanisms is the neural threshold. There are several experimental observations and hypotheses about the possible computational properties that the neural threshold dynamics could control, like its role in ensuring that the neural code is optimal and its influence on the process of discrimination of stimuli in sensory systems.

Some studies carried out on the olfactory system of insects by the Grupo de Neurocomputación Biológica at UAM suggests that a heterogeneous distribution of neural thresholds among the Kenyon cells in the mushroom body is one of the factors that regulates the excitability of the neurons and enables the emergence of the sparse code used by this system to represent the information and improve its ability to discriminate odorants.

Continuing with this approach, the goal that has been proposed in this TFM is the development of a model of the olfactory system of the insect that allows to explore the influence of the distribution of neural thresholds in its discrimination capacity, the emergence of sparse code and its energy efficiency.

To achieve this goal, a new model of the locust olfactory system has been developed. The model is based on neural networks and supervised learning and includes a learning algorithm capable of adjusting the firing thresholds of KCs. In addition, the algorithm also has a regulation term to control the level of activation of KCs.

To test the performance of the proposed model, three datasets with patterns of different levels of complexity have been used. The main conclusion that has been obtained is that the model learning algorithm is able to regulate the activity of KCs through the adjustment of their neural thresholds and obtains better results when it converges to a state similar to the one found in biology. In addition, it is also able to adjust the characteristics of the distribution of thresholds in the KCs to adapt it to the complexity of the patterns that are being classified. In order to achieve this, the algorithm sets different balances between the number of neurons with a high threshold and specialist behavior and a low threshold with generalist behavior. This is consistent with other studies conducted in the GNB on the influence of the balance between specialist and generalist neurons in the performance for a pattern recognition task. Therefore, the results obtained show that the neuronal threshold is of great importance when determining the discrimination capacity of the system and controls the generation of sparse code for the representation of information.

Key words

Computational neuroscience, olfactory system, threshold potential, Kenyon cells, bio-inspired neural networks, machine learning, biological systems modeling, pattern recognition, neural heterogeneity, sparse code, energy efficiency

Agradecimientos

Agradezco sinceramente a mis padres y amigos el haberme apoyado durante esta etapa académica. Gracias también a Francisco B. Rodríguez y Aarón Montero por su orientación y dedicación durante la realización de este trabajo.

Índice general

Índice de Figuras

Índice de Tablas

1 Introducción

En esta sección se realiza una introducción al proyecto, explicando su motivación y enumerando los objetivos que se quieren conseguir. También se describe brevemente la estructura de este documento y el contenido de cada uno de sus capítulos.

1.1. Motivación

Los sistemas biológicos de procesamiento de información codifican la misma mediante impulsos eléctricos conocidos como disparos neuronales. Los disparos neuronales se producen por la apertura y el cierre de ciertos canales iónicos de la membrana celular que modifican su potencial durante un corto intervalo de tiempo [1, 2]. Existen al menos cuatro tipos de codificación neuronal, conocidos como tasa de disparos neuronal, codificación temporal, codificación poblacional y codificación dispersa. Estos tipos de codificación no aparecen de forma independiente, sino que pueden combinarse entre sí, pero todos ellos se basan en la variación de las características del proceso de generación de disparos [3] y han sido y siguen siendo objeto de muchos estudios, tanto mediante modelos matemáticos como mediante observaciones empíricas directamente sobre los sistemas biológicos, con el objetivo de determinar sus capacidades computacionales.

Por otro lado, las investigaciones sobre los mecanismos subyacentes que controlan el proceso de generación de disparos también son esenciales para comprender cómo se codifica la información y qué factores controlan la aparición de los diferentes tipos de código. Uno de estos mecanismos de regulación de la generación de impulsos es el umbral neuronal, que se define como el potencial que la membrana de la neurona tiene que superar para que se produzca un disparo. Dependiendo del valor que tome el umbral, se producirán más o menos disparos en el tiempo con una determinada distribución y, por tanto, se modulará la codificación de la información [4].

Teniendo en cuenta lo anterior, la dinámica del umbral neuronal debe determinar en gran medida la relación entre la señal de entrada y la salida de las neuronas, por lo que existen existen numerosas observaciones empíricas e hipótesis sobre los diferentes propósitos de estos umbrales en sistemas biológicos. En este contexto, algunas investigaciones llevadas a cabo por el Grupo de Neurocomputación Biológica (GNB) apuntan a que una distribución heterogénea en los umbrales de ciertas poblaciones neuronales del sistema olfativo de los insectos es uno de los factores que regula la excitabilidad de las neuronas y posibilita la aparición del código disperso empleado por este sistema, mejorando así su capacidad de discriminación de estímulos [5, 6, 7].

Dentro de esta línea de investigación, el objetivo de este Trabajo de Fin de Máster es el desarrollo de un modelo del sistema olfativo del insecto que permita explorar la influencia de la distribución de umbrales neuronales en la capacidad de discriminación del sistema, la generación del código disperso y su eficiencia energética [8, 9, 10], con el fin último de determinar qué propiedades computacionales posibilita y controla el umbral neuronal. El modelo propuesto en este trabajo se basa en redes neuronales y es capaz de ajustar, mediante un algoritmo de aprendizaje diseñado para tal efecto, la distribución de umbrales de las neuronas de la red para un problema de clasificación determinado. El modelo y los resultados preliminares que se han obtenido a partir de él se han presentado en los congresos CNS18 [11] e ICANN18 [12].

1.2. Objetivos y enfoque

Este TFM se centra en estudiar la influencia del umbral neuronal en la aparición de código disperso, la eficiencia energética y la capacidad de discriminación de estímulos en el sistema olfativo de los insectos. Para ello, se plantean los siguientes objetivos:

- Estudiar las diferentes hipótesis que se han propuesto sobre las propiedades computacionales del umbral neuronal.
- Estudiar los diferentes modelos del sistema olfativo de los insectos que se han utilizado previamente para analizar su comportamiento y estimar el valor de algunos de sus parámetros más importantes.
- Desarrollar un modelo del sistema olfativo basado en redes neuronales y aprendizaje supervisado capaz de encontrar la distribución de umbrales óptima para un conjunto de patrones de clasificación. Esto nos permitirá investigar las propiedades de la distribución de umbrales, su influencia en la aparición del código disperso y en el control del nivel de actividad de las neuronas.
- Evaluar la capacidad de discriminación del modelo desarrollado con diferentes problemas de clasificación para determinar qué combinaciones de valores de sus parámetros permiten que alcance el error mínimo.
- Comparar los resultados obtenidos con lo observado en los sistemas biológicos y con los resultados de las investigaciones previas llevadas a cabo en el GNB.

En cuanto a la implementación del modelo, se ha elegido Python como lenguaje de programación ya que cuenta con numerosas librerías y facilidades para realizar tareas de análisis de datos, aprendizaje automático y representación, tales como Numpy, Scipy, Scikit-learn y Matplotlib. Las simulaciones del modelo con diferentes conjuntos de parámetros se llevaron a cabo en el *cluster* de computación que posee el GNB.

1.3. Estructura de la memoria

Esta memoria se organiza en los siguientes capítulos:

- **Estado del arte:** en esta sección se exponen los conceptos previos necesarios para la comprensión del trabajo realizado, como los diferentes estudios llevados a cabo sobre las propiedades computacionales del umbral neuronal, el sistema olfativo de la langosta voladora y la modelización de sistemas biológicos.

- **Diseño y desarrollo:** en este capítulo se exponen los detalles del modelo de sistema olfativo que se ha desarrollado, como su estructura, las estrategias que incluye para el reconocimiento de patrones y su algoritmo de aprendizaje. También se explican las características de los diferentes problemas de clasificación que se han elegido para realizar la evaluación del modelo.
- **Experimentos y análisis de resultados**: en esta sección se muestran los resultados obtenidos a partir de las simulaciones del modelo de sistema olfativo con varias combinaciones de sus parámetros para los diferentes problemas de clasificación seleccionados. Estos resultados incluyen el error de clasificación conseguido por el modelo, la actividad media de sus neuronas y las características de las distribuciones de umbrales obtenidas, así como la discusión de los resultados.
- **Conclusiones y trabajo futuro**: se exponen las conclusiones alcanzadas con este proyecto y cómo se va a seguir desarrollándo en el futuro.
- **Anexos:** figuras adicionales e información sobre las herramientas software desarrolladas.

2

Estado del arte

2.1. Introducción

En esta sección se exponen los conceptos y conocimientos previos necesarios para comprender este proyecto, tales como las diferentes hipótesis que se han propuesto sobre las propiedades computacionales del umbral neuronal y del código disperso, el funcionamiento y estructura del sistema olfativo del insecto y las técnicas utilizadas para el modelado de sistemas biológicos de procesamiento de información, así como los diferentes modelos del sistema olfativo de los insectos que se han propuesto con anterioridad.

2.2. Propiedades computacionales del umbral de disparo

El umbral de disparo neuronal se define como el voltaje que el potencial de la membrana de la neurona tiene que superar para que se produzca un disparo neuronal por medio de la apertura y el cierre de los canales iónicos en la membrana. En la Figura 2.1. se puede observar una representación gráfica de un disparo neuronal y el umbral de disparo. Aunque la visión clásica establece que el umbral siempre toma un valor más o menos constante entre -55mV y -45mV [1, 2], esto no es suficiente para explicar algunos fenómenos observados en todos los sistemas biológicos como es, por ejemplo, la adaptación de las neuronas ante estímulos constantes mediante un aumento de su refractariedad [13, 14]. Para explicar estos comportamientos, se propuso la teoría de que el umbral de disparo varía con el tiempo en función de la señal de entrada que la neurona recibe. Por tanto, la existencia de una dinámica del umbral neuronal determina de forma importante la codificación de la información. Las propiedades computacionales que podría introducir y controlar siguen investigándose y en la actualidad existen diferentes hipótesis basadas en observaciones experimentales en sistemas biológicos diferentes sobre cuál es su función computacional dentro del sistema. A continuación se presentan algunas de las más importantes:

- **Regulación de neuronas especialistas/generalistas:** en algunos sistemas sensoriales como el sistema olfativo de los insectos se han observado grupos de neuronas que reaccionan de forma diferente ante los estímulos. Algunas de ellas, conocidas como generalistas, poseen una mayor frecuencia de disparo al ser sensibles a la mayoría de los estímulos, mientras

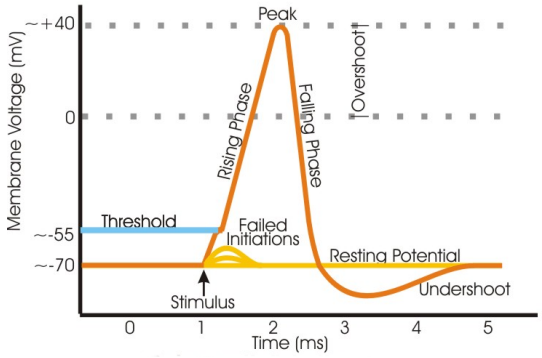

Figura 2.1: Potencial en la membrana frente al tiempo durante un disparo neuronal. Cuando el potencial de membrana supera el valor del umbral, se produce el disparo neuronal. Los incrementos del potencial que no alcanzan el valor del umbral no producen disparos y regresan al potencial de reposo con el tiempo. Figura realizada por Synaptidude en en.wikipedia [GFDL (http://www.gnu.org/copyleft/fdl.html) o CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0/)], vía Wikimedia Commons.

las especialistas apenas muestran actividad ya que sólo responden ante un conjunto muy reducido de estímulos. Este comportamiento generalista/especialista estaría regulado por una distribución heterogénea del valor del umbral de disparo en la población de neuronas y contribuiría a mejorar el reconocimiento de patrones y la identificación de los estímulos [7, 6, 5, 15, 16].

- **Selectividad de atributos de los estímulos:** en los sistemas cuyo propósito es analizar información sensorial, como el córtex visual o el auditivo, existen neuronas cuya respuesta muestra selectividad ante determinados atributos de los estímulos que les llegan, como puede ser el ángulo en el que se mueva un objeto en el campo visual, siendo su respuesta más o menos fuerte en función de su valor. Este fenómeno fue descrito hace más de 20 años [17], pero no se profundizó en las causas subyacentes al mismo. Una hipótesis planteada para explicarlo propone que este comportamiento estaría regulado por las dinámicas del umbral neuronal, ya que se ha observado que los umbrales de disparo en estos sistemas aumentan o disminuyen dependiendo del valor del atributo al que son sensibles y de sus valores previos, regulando la intensidad de la respuesta neuronal [18, 19, 20, 15].

- **Eficiencia energética:** la generación de disparos neuronales es un proceso energéticamente muy costoso y, por tanto, tras millones de años de evolución, es de esperar que el código neuronal se haya optimizado de tal forma que se minimice el gasto energético a la vez que se maximiza la cantidad de información transmitida. Uno de los mecanismos que se proponen para el control y el mantenimiento de la eficiencia energética es el umbral neuronal. Al depender el umbral de la evolución de la señal de entrada, puede regular el nivel de excitabilidad de las neuronas a largo plazo, así como limitar el número de disparos cuando hay ruido o el estímulo se mantiene constante durante largos periodos de tiempo. Por otro lado, también contribuye a la eficiencia energética la fluctuación del umbral a valores más altos cuando la tasa de disparo es alta y a valores más bajos cuando el sistema presenta poca actividad. De esta forma, el umbral neuronal contribuiría a la hora de controlar el consumo de energía y la calidad de la información transmitida [21, 22].

- **Memoria a corto plazo:** esta hipótesis se ha plantado a raíz de algunos estudios sobre el hipocampo. En este sistema, las neuronas repiten secuencias de disparo que perduran en el tiempo y, por tanto, es necesario que cuente con alguna representación del tiempo transcurrido. Al depender el umbral de disparo de las características anteriores de la señal

de entrada y de los disparos que se han ido produciendo, se plantea que podría funcionar como una memoria a corto plazo básica. De esta forma no sería necesario introducir otros elementos más complejos como patrones de conectividad complicados o aprendizaje de secuencias para que las neuronas mostrasen secuencias de disparos recurrentes en el tiempo[23, 24].

- **Filtrado y sincronización:** el umbral neuronal también podría funcionar como un filtro que se aplica sobre la señal de entrada de la neurona y permite realizar una codificación temporal de la evolución de dicha señal con gran precisión, en algunos casos, pudiendo llegar a sincronizar los disparos con sus fluctuaciones. Por otro lado, dado que el umbral puede modelar la respuesta de la neurona en función de la magnitud y la velocidad de la despolarización del potencial de membrana, puede actuar como un detector de coincidencias que amplifica la llegada de potenciales coincidentes de otras neuronas y descarta las entradas asíncronas [25, 26, 27, 28, 29].
- **Codificación robusta:** la dinámica del umbral de disparo dotaría a la codificación neuronal de mayor robustez y tolerancia hacia errores. De esta forma, el umbral contribuiría a reducir el efecto del ruido o generar las mismas secuencias de disparo independientemente del estado inicial del sistema [30, 29]. Dentro de esta línea, el umbral también podría contribuir a mantener acotado el error producido durante la codificación de la información. La representación que la neurona hace de la entrada que le llega sería filtrada mediante el umbral y comparada con la señal original. Cada vez que el error entre ambas señales superase cierto valor, se produciría un disparo para modificar el valor del umbral y reducir el error [21]. Sin embargo, este último planteamiento aún necesita de muchas más evidencias biológicas y comprobaciones para poder considerarse.

Las hipótesis que se han presentado están muy interrelacionadas entre sí, ya que todas ellas se basan en propiedades emergentes del umbral de disparo que surgen como consecuencia de su comportamiento y su relación con la evolución del potencial de membrana de la neurona en el tiempo. Por tanto, estas hipótesis podrían constituir mecanismos generales en los sistemas biológicos para codificar y procesar la información. Este trabajo se centra en el papel del umbral neuronal en el sistema olfativo de los insectos y explora las hipótesis que presentan al umbral neuronal como un mecanismo regulador del balance entre neuronas generalistas y especialistas, la eficiencia energética, la codificación robusta y su función como filtro y detector de coincidencias.

2.3. Código disperso

Diversos estudios han mostrado que en algunos sistemas biológicos la información se codifica mediante grupos de neuronas que muestran una actividad muy baja y una gran selectividad ante los estímulos a los que responden, por lo que la mayor parte del tiempo no muestran actividad en forma de disparos. Este tipo de comportamiento se conoce como código disperso y podría tratarse de una estrategia general de codificación eficiente empleada por los sistemas biológicos [31, 3].

El código disperso se ha observado, sobre todo, en las últimas capas de los sistemas que procesan la información de estímulos sensoriales y sus ventajas han sido analizadas por diversos estudios teóricos y modelos computacionales [31]. Entre los sistemas biológicos que utilizan este tipo de codificación se encuentran el sistema olfativo de la langosta voladora [10, 32, 9] (en el que se centra este trabajo), el córtex visual [33, 34], el sistema auditivo [35, 36] y el hipocampo [37, 38, 39].

Entre las ventajas que tiene la codificación dispersa, destacan que su uso puede aumentar la capacidad de las memorias asociativas [40] y puede facilitar el aprendizaje y la clasificación de

patrones en presencia de ruido [9]. Estas ventajas se consiguen por medio del uso de un código sobre-completo para representar los estímulos. Este tipo de código se caracteriza por tener una dimensionalidad mucho mayor a la del espacio de entradas que codifica y, por tanto, es capaz de representar un número mayor de entradas que las necesarias, por lo que será difícil que se produzcan colisiones entre diferentes representaciones.

En los sistemas biológicos, la codificación sobre-completa se consigue mediante el uso de una población neuronal con una dimensionalidad mucho mayor que los estímulos que va a representar. De esta forma, cada neurona podrá tener un subconjunto de estímulos muy reducido hacia los que muestra preferencia y se activará sólo cuando la entrada se parezca lo suficiente a alguno de ellos. Esto hace que sea mucho más fácil la detección de patrones y la creación de asociaciones mediante reglas de aprendizaje locales. Por tanto, debe establecerse un balance en el sistema entre las ventajas de utilizar un espacio de mayor dimensionalidad para codificar estímulos y el coste por tener que mantener un número mayor de neuronas en la población [31, 41, 32, 8].

Por último, otro factor de gran importancia a la hora de favorecer la aparición de código disperso es su eficiencia energética [31]. Como ya se señaló en la Sección 2.3, la generación de disparos neuronales es un proceso muy costoso energéticamente y es de esperar que se haya optimizado para minimizar el consumo y maximizar la tasa de información codificada [21]. En este sentido, el código disperso cumpliría ambas condiciones y su uso se habría visto favorecido por esta razón.

2.4. Sistema olfativo de la langosta voladora

Dada la complejidad de los sistemas biológicos de procesamiento de la información, es común estudiar organismos sencillos como insectos y pequeños mamíferos en los que, a pesar de su capacidad, la complejidad de las redes neuronales que intervienen es más reducida y los resultados potencialmente generalizables a redes más complejas. Uno de los sistemas biológicos más estudiados por ser de menor complejidad que los de los vertebrados es el sistema olfativo de los insectos, cuya estructura y algunas de las funciones que realizan diferentes poblaciones de neuronas son conocidas. En concreto, este trabajo se centra en el sistema olfativo de la langosta voladora.

El sistema olfativo de la langosta se organiza en diferentes capas de procesamiento. En primer lugar, las neuronas receptoras olfatorias (ORNs) situadas en las antenas del insecto captan la información de los odorantes presentes en el aire. Cada ORN es receptiva a un conjunto concreto de odorantes [42]. La información sobre los odorantes captada por las ORNs pasa a la siguiente capa de procesamiento: el lóbulo antenal (AL). En el lóbulo antenal se encuentran los glomérulos olfatorios, las neuronas de proyección (PNs) y las inter-neuronas locales (LNs). Estas últimas pueden ser tanto excitatorias como inhibitorias. Cada glomérulo recibe la información de un conjunto de ORNs que son sensibles al mismo rango de olores, creándose así un canal de información para cada tipo de olor [42, 43]. Los glomérulos envían su salida tanto a las LNs como a las PNs. Aunque las ORNs son sensibles ante la concentración de los odorantes y presentan mayor actividad cuanto mayor es la concentración [44], la tasa de disparo en las PNs se mantiene más o menos constante [45] con independencia del nivel de excitación que reciban de los glomérulos debido a la inhibición producida por las LNs. Por tanto, las LNs funcionan como un mecanismo de control de ganancia para modular la respuesta de las PNs, de forma que los olores puedan ser representados de manera estándar con independencia de su concentración y, por tanto, hacer que sean más fácilmente identificables [43, 46, 47]. La respuesta de las PNs se caracteriza por ser oscilatoria, donde periodos de actividad en los que se producen ráfagas de disparos se alternan con otros de relativo silencio [10, 32, 48]. Esta actividad se proyecta al cuerpo fungiforme, la siguiente capa en el procesamiento.

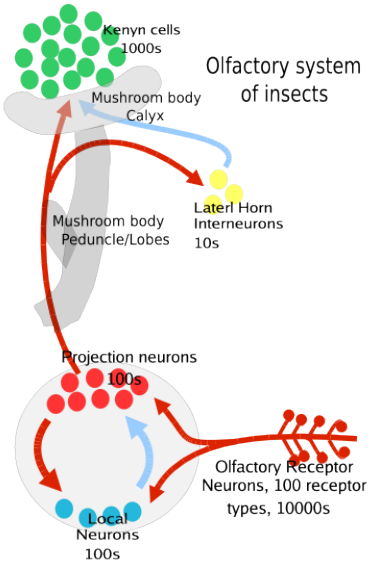

Figura 2.2: Esquema del sistema olfativo de un insecto. Figura de Badjoby commonswiki [CC BY-SA 2.5 (https://creativecommons.org/licenses/by-sa/2.5)], vía Wikimedia Commons

En el cuerpo fungiforme se encuentran las células de Kenyon (KCs) y las neuronas de salida (MBONs). Las KCs reciben la información del lóbulo antenal mediante sus conexiones con las PNs. El patrón de conectividad entre ambos tipos de neuronas no muestra reproducibilidad entre individuos y tampoco se produce aprendizaje. La probabilidad de que exista una conexión entre una KC y una PN ha sido objeto de gran controversia durante años, pero recientemente se ha establecido que este valor estaría en torno a 0.5 [9, 49, 50].

Las KCs se caracterizan por presentar una actividad muy baja, ya que solo responden mediante disparos neuronales a conjuntos de odorantes muy concretos [10, 9, 32], mientras el resto del tiempo se mantienen en silencio o respondiendo a los estímulos sólo mediante oscilaciones sub-umbrales [9]. Por tanto, las KCs utilizan un código disperso para representar la información que les llega del lóbulo antenal y funcionan como detectores de coincidencia que sólo responden cuando reciben suficiente estimulación de un grupo de PNs muy concreto [32, 51]. En este sentido, el umbral de disparo de las KCs, definido como la cantidad de PNs de las que deben recibir excitación para producir un disparo, sería un parámetro de gran importancia a la hora de regular la actividad del sistema [9].

Uno de los factores que controlan el grado de excitabilidad de las KCs es la inhibición producida por neuronas como el gigante GABAérgico (GGN) [52, 53]. Se cree que esta neurona recibe como entrada el nivel de actividad de todas las KCs e inhibe a todas ellas de forma proporcional, por lo que contribuye a regular la salida de las KCs y mantener el código disperso [32, 53, 51]. La acción de la GGN junto con el carácter oscilatorio de la excitación que reciben las KCs por parte de las PNs modelan su actividad de forma cíclica: las KCs reciben excitación de las PNs y al aumentar su actividad son inmediatamente inhibidas por la acción de la GGN. Por tanto, la respuesta en las KCs sólo puede producirse en el periodo entre inhibiciones sucesivas por parte de la GGN. La longitud de este periodo depende del grado de excitación transmitido por las PNs, ya que un nivel mayor o menor de actividad provoca que la GGN responda antes o después [32, 54]. Además de lo anterior, otros factores que contribuyen a la generación del código disperso en la población de KCs son las propiedades fisiológicas de estas neuronas. En su membrana celular, las KCs poseen canales iónicos con valores de conductancia que difieren de los encontrados en otras neuronas, lo que les ayudaría a amplificar las entradas coincidentes más fuertes de otras neuronas y a adaptarse rápidamente a los estímulos por medio del aumento del valor de su umbral de disparo [55].

Las ventajas que el código disperso tiene en este sistema están en la línea que se describió en la Sección 2.3. Por un lado, dado que el cuerpo fungiforme funciona como una memoria asociativa, mejora la capacidad de la misma [32, 41]. Además, el ratio entre la población de PNs y la de KCs en la langosta es de 1:50. Este ratio supone que la codificación de los olores realizada por las PNs se proyecta en un espacio de mucha mayor dimensionalidad, lo que, junto al código disperso, favorece la separabilidad de los patrones y la ausencia de colisiones entre sus representaciones [41, 8].

Por último, la asociación entre las representaciones realizadas por las KCs y la identidad de cada olor se realiza mediante aprendizaje en las sinapsis entre las KCs y las MBONs. Entre las MBONs existe inhibición mutua de forma que sólo se activa un subconjunto de las mismas para cada odorante identificado [41, 8].

Este trabajo se centra sobre todo en el cuerpo fungiforme y las KCs con el fin de estudiar cómo se controla el nivel de actividad de las KCs a través de la regulación del umbral neuronal para generar el código disperso utilizado por esta capa.

2.5. Modelos computacionales del sistema olfativo

En esta sección se describen algunos conceptos básicos sobre la necesidad de realizar modelos computacionales para el estudio de sistemas biológicos, las diferentes técnicas que existen para su diseño e implementación y, por último, los princpales modelos computacionales del sistema olfativo de los insectos que se han propuesto para analizar su comportamiento.

2.5.1. Modelado de sistemas biológicos

La gran complejidad de los sistemas biológicos de procesamiento de información y el hecho de que solo se disponga de datos parciales sobre su funcionamiento, ya que es no es posible observarlos de forma completa, hacen necesario el diseño y la implementación de modelos que intenten reproducir su estructura y propiedades. El objetivo de los modelos de sistemas biológicos es poder facilitar la tarea de plantear y comprobar hipótesis sobre su funcionamiento y analizar la influencia de los diferentes parámetros en el sistema. Con las conclusiones obtenidas a partir de los modelos, pueden diseñarse nuevos experimentos que permitan contrastar los resultados mediante observaciones y datos empíricos. De esta forma, los modelos contribuyen a facilitar la comprensión de las estrategias para el procesamiento de la información que utilizan los sistemas biológicos [56].

El procesamiento de la información en los sistemas biológicos se lleva a cabo en diferentes niveles: subcelular, celular, a nivel de red y a nivel del sistema completo. Estos niveles tienen escalas espaciales y temporales diferentes cuya integración da lugar al procesamiento de la información. Dependiendo del nivel de la escala de procesamiento que se quiera estudiar, el modelo deberá incluir más o menos detalles, elementos o grado de realismo, por lo que existen diferentes técnicas para su implementación [56].

Modelado de neuronas y sinapsis

En la mayoría de modelos de sistemas biológicos, se toma la neurona como la unidad básica de procesamiento de la información y, por tanto, para construir un modelo de un sistema, en primer lugar debe elegirse cómo se van a representar las neuronas. A lo largo de los años se han desarrollado una gran cantidad de modelos neuronales con diferente nivel de detalle y coste

computacional, de forma que se pueda seleccionar el que mejor se adapte a las características y restricciones del modelo que se va a implementar [56].

Uno de los primeros y más detallados modelos que se han propuesto es el modelo de Hodgkin-Huxley [57]. Este modelo tiene en cuenta las dinámicas de activación y desactivación de los diferentes canales iónicos que producen los disparos neuronales mediante varias ecuaciones diferenciales y variables que modelan su comportamiento. Es capaz de generar diversos patrones de actividad neuronal casi idénticos a los que se observan en las neuronas de muchos sistemas biológicos y, por tanto, constituye una opción muy potente para modelos que necesitan mucho realismo. Sin embargo, una de sus principales desventajas es que requiere de un elevado coste computacional que hace imposible su uso para modelar sistemas más complejos que incluyen un número elevado de neuronas.

Para poder contar con modelos neuronales capaces de imitar el comportamiento de las neuronas biológicas sin un excesivo coste computacional, se han desarrollado diversos modelos simplificados basados en el modelo de Hodgkin-Huxley pero reduciendo el número de ecuaciones utilizadas. Dentro de estos modelos se encontrarían el de Hindmarsh-Rose [58] o los modelos de integración y disparo [59]. También existen modelos más abstractos que consideran a las neuronas como integradores de potenciales que disparan por encima de un umbral, como el modelo de McCulloch-Pitts [60]. De esta forma, se establece un equilibrio entre el realismo del modelo y la complejidad de los cálculos necesarios para implementarlo.

El modelo de McCulloch-Pitts fue una de los primeros modelos computacionales de neuronas propuestos, y se define como:

$$V(t+1) = H(I(t) - \theta) \, , \tag{2.1}$$

donde $V(t)$ representa el potencial de membrana de la neurona en el tiempo t, $I(t)$ representa la suma de las corrientes de entrada procedentes de otras neuronas, θ es el umbral de disparo de la neurona y $H(x)$ es la función *Heavisde* [60]. El modelo de McCulloch-Pitts se ha utilizado en numerosas ocasiones para modelar el comportamiento de las neuronas KCs, ya que estas funcionan como integradores de los disparos que reciben de las PNs y solo disparan si la entrada que reciben es suficientemente fuerte para superar su umbral de disparo. Para modelar este comportamiento, en principio tampoco es necesario tener en cuenta la escala temporal en la que se producen los disparos, por lo que el modelo de McCulloch-Pitts es adecuado [8, 41, 5, 6].

Otro aspecto fundamental para el modelado de sistemas biológicos son las sinapsis neuronales. Las sinapsis son el mecanismo de comunicación entre neuronas, a través del cual se transmiten los impulsos eléctricos. Las sinapsis pueden ser de diferentes tipos dependiendo del efecto que tengan en la neurona postsináptica. Así, pueden ser inhibitorias, que depolarizan su membrana haciendo que sea más difícil que se produzcan disparos, o excitatorias, que tienen el efecto inverso. Normalmente, esta propiedad se modela mediante el signo de la corriente que recibe la neurona. Por otro lado, según el mecanismo que se utiliza para la transmisión, las sinapsis pueden ser eléctricas o químicas.

Las sinapsis eléctricas se caracterizan por ser más rápidas que las químicas y se llevan a cabo mediante uniones *gap* entre las neuronas. Cuando los potenciales eléctricos llegan a la unión gap, hiperpolariza o depolariza la membrana de la neurona postsináptica [1, 2]. Este tipo de sinapsis puede modelarse fácilmente multiplicando la diferencia de potencial entre ambas neuronas por un valor de conductancia, de forma que los potenciales de ambas neuronas se sincronicen con mayor o menor rapidez.

Las sinapsis químicas se llevan a cabo mediante neurotransmisores. Cuando un potencial llega al punto de unión entre las neuronas, produce la apertura de una serie de vesículas. Estas vesículas liberan cierta cantidad de neurotransmisores en el espacio existente entre las membranas de las dos neuronas. Los neurotransmisores se unen a sus receptores específicos en la

membrana de la neurona postsináptica y provocan la apertura de ciertos canales iónicos que depolarizan o polarizan la membrana. Estas sinapsis se modelan mediante diferentes ecuaciones y parámetros en función del tipo de neurotransmisor por el que estén mediadas y la escala temporal en la que se encuentre en relación con el resto del sistema. En [61] se puede consultar una aproximación general para modelar diferentes tipos de sinapsis químicas. Normalmente, las sinapsis excitatorias se realizan mediante el neurotransmisor glutamato, cuyos receptores son AMPAR [62] y NMDAR [63], mientras las sinapsis inhibitorias se realizan mediante GABA y sus receptores GABARa [63] y GABARb [1, 56].

Mediante los modelos existentes para simular el comportamiento de neuronas y sinapsis puede estudiarse las características muchas redes neuronales de tamaño reducido o simplificados, como pueden ser determinados circuitos dentro de sistemas de invertebrados o pequeños mamíferos. Sin embargo, para redes completas de mayor escala como el sistema olfativo de la langosta que consta de decenas de miles de neuronas, el coste computacional y la complejidad de estos modelos hace que su uso sea inviable, por lo que es necesario el uso de otras técnicas que se describirán a continuación.

Modelado de sistemas biológicos mediante redes neuronales

Las redes neuronales interpretan los sistemas biológicos como redes formadas por nodos que representan las neuronas y conexiones entre ellas que representan las sinapsis neuronales. Cada nodo tiene asociado un valor, que representar su potencial de membrana, que se calcula como la suma de las sinapsis recibidas. Cada sinapsis tiene además un valor asociado, que representa la fuerza de la conexión, y que puede ser excitatoria o inhibitoria. Las redes neuronales han sido de gran utilidad tanto para el modelado de sistemas biológicos como para aplicaciones de aprendizaje automático. Su éxito se debe a que su estructura permite que emerjan propiedades similares a las de las redes neuronales biológicas, como reconocimiento de patrones, memoria asociativa o aprendizaje [56]. En este trabajo nos centraremos en las redes neuronales para reconocimiento de patrones, ya que el sistema olfativo de los insectos realiza esta tarea con los odorantes que capta.

El aprendizaje de patrones en las redes neuronales se lleva a cabo mediante el ajuste del peso asociado a cada una de las sinapsis de la red. Este aprendizaje puede ser supervisado o no supervisado. En el aprendizaje supervisado se da a la red información sobre la clase de los patrones que se le están presentando, de forma que puede ajustar el valor de sus diferentes parámetros en función del error que comete a la hora de hacer sus predicciones. En este tipo de aprendizaje, los algoritmos que suelen utilizarse se basan en el método matemático de descenso por gradiente para una función de coste, como en el caso del algoritmo de retropropagación [64]. Por otro lado, en el aprendizaje no supervisado no se tiene información sobre las clases a las que pertenecen los patrones, por lo que la red debe ser capaz de extraer la información relevante de los patrones y autorganizarse para generar una clasificación satisfactoria. Para ello se suelen aplicar reglas de aprendizaje asociativo como las de Hebb [64]. Mediante el uso de estas reglas, las sinapsis entre neuronas se refuerzan o se atenúan en función de los patrones de disparo de las neuronas entre las que existe una conexión. Dado que los sistemas biológicos se caracterizan por su autorganización y por no tener información externa acerca de los patrones, el aprendizaje no supervisado se suele aplicar con más frecuencia al modelado de sistemas biológicos, aunque el aprendizaje supervisado también puede suponer ventajas en casos concretos.

Por último, entre las redes neuronales artificiales más estudiadas y utilizadas se encuentra el perceptrón multicapa. Este tipo de red fue propuesta tras comprobar que el perceptrón simple no podía resolver problemas no lineales. El perceptrón multicapa hace uso del algoritmo de retropropagación y de funciones de activación no lineales para resolver este tipo de problemas y constituye un aproximador universal, capaz de converger a cualquier función que exista entre

la entrada y la salida proporcionadas [64]. El perceptrón multicapa está formado por tres tipos de capas diferentes. La capa de entrada se corresponde con los patrones que llegan a la red; las capas ocultas reciben su entrada de capas anteriores y proyectan su salida a la capa siguiente; la capa de salida se corresponde con la clasificación que predice la red para un patrón. Durante el aprendizaje, la predicción de la capa de salida se compara con la predicción correcta y el error cometido se propaga hacia las capas anteriores, actualizando los pesos de las sinapsis en consecuencia para reducir el error cometido. Para modelizar el sistema olfativo de la langosta, en este trabajo se partirá de un perceptrón con una sola capa oculta sobre el que se introducirán diversas modificaciones en su estructura, topología y algoritmo de aprendizaje para implementar un modelo del sistema olfativo de la langosta. Este modelo se utilizará para explorar la influencia del umbral neuronal y el código disperso en el reconocimiento de patrones.

2.5.2. Modelos del sistema olfativo de los insectos

En esta sección se describen algunos de los modelos del sistema olfativo de los insectos que se han propuesto y utilizado en otros trabajos sobre este sistema para estudiar sus propiedades computacionales.

Modelos probabilísticos del sistema olfativo

Los modelos que se presentan en este apartado están basados en probabilidades y se desarrollan en [8, 41]. En estos trabajos se propone que mecanismos sencillos como una conectividad aleatoria entre las PNs y las KCs, aprendizaje Hebbiano en el MB, inhibición mutua ente las MBONs y la estructura de cada capa son suficientes para explicar el complejo reconocimiento de patrones llevado a cabo en el sistema olfativo de la langosta. La hipótesis planteada es que el sistema realiza una transformación no lineal de los patrones de actividad de las PNs a la capa de las KCs proyectándolos en un espacio de mucha mayor dimensionalidad, de forma que a cada patrón de activación de PNs le corresponde un único patrón en las KCs. A partir de esto, las KCs se modelan como unidades de McCulloch-Pitts y se desarrolla una medida de inyectividad, $I \in [0{,}0, 1{,}0]$, de esta función de transformación. Para un correcto funcionamiento del sistema, I debe tener un valor próximo a 1,0 para que no se produzcan colisiones y la identificación de patrones sea óptima, lo que está en consonancia con las ventajas que el código disperso supone. Uno de los parámetros más influye en el valor de I en el sistema es el umbral de disparo de las neuronas KCs, así como la probabilidad de conexión p_c entre las PNs y las KCs. Los valores óptimos de ambos parámetros para asegurar un valor de I próximo a 1 y la eficiencia energética del sistema son $\theta_{KC} \in [8, 11]$ y $p_c \in [0{,}015 - 0{,}025]$, aunque los valores de p_c que se proponen están muy por debajo de los asumidos en estudios recientes ($p_c \approx 0{,}5$). Una vez realizada la representación de los estímulos en las KCs, el aprendizaje en las sinapsis entre las KCs y las MBONs, que puede modelarse mediante aprendizaje Hebbiano, relaciona los diferentes patrones de activación de las MBONs con los diferentes olores. Esto, junto con la inhibición mutua entre las MBONs, es responsable del aprendizaje y el reconocimiento de los estímulos. Otra de las hipótesis que se plantean es que el nivel de actividad en las KCs está relacionado con la redundancia que el sistema debe introducir para solventar los errores de transmisión de la información que se puedan producir y, por ello, la actividad en esta capa sería ligeramente superior a lo que podría esperarse.

Lo importante a destacar de estos estudios en relación con el trabajo que se va a desarrollar en este proyecto es que se muestra que el umbral neuronal de las KCs es uno de los parámetros del sistema claves a la hora de controlar la actividad en las KCs, el código disperso y la capacidad de discriminación del sistema. Sin embargo, también es importante destacar que en estos trabajos no se tiene en cuenta el papel que el control de ganancia puede jugar en la representación de

los patrones y la capacidad de discriminación de la red ni la inhibición que reciben las KCs por parte de la GGN, aunque esto último sí se incluye aunque de forma indirecta mediante la regulación del umbral neuronal de las KCs.

Modelos del sistema olfativo basados en redes neuronales

A partir de las conclusiones obtenidas en los análisis teóricos de los modelos del apartado anterior y los resultados del el análisis de datos procedentes de mediciones neurofisiológicas realizadas en el sistema olfativo biológico de la langosta [10, 65, 9], en [5, 6, 7] se propone y desarrolla un modelo del sistema olfativo basado en redes neuronales. Este modelo se emplea para contrastar la hipótesis de que la heterogeneidad en los sistemas biológicos favorece su funcionamiento y, en concreto, se propone que una distribución de umbrales heterogénea en las KCs modula el nivel de excitabilidad de cada neurona de tal forma que surgen dos poblaciones diferentes: una de neuronas especialistas que sólo responden hacia unos pocos estímulos y otras KCs con un comportamiento maś generalista que disparan ante un rango mayor de estímulos sin alterar la naturaleza del código disperso empleado por esta capa del sistema. Esta idea estaría de acuerdo con las conclusiones obtenidas tras el análisis de los datos de las KCs presentado en [65].

El modelo del sistema olfativo planteado en [5, 6, 7] se basa en una red neuronal con una sola capa oculta. La entrada de dicha red se corresponde con las PNs, la capa oculta con las KCs y la capa de salida con las MBONs. Esta red implementa los mecanismos sencillos planteados en [41], como el modelado de las KCs como unidades McCulloch-Pitts, la conexión aleatoria con cierta probabilidad p_c entre las PCs y las KCs, el aprendizaje Hebbiano y la proporción 1:50 entre neuronas PNs y KCs. El modelo se emplea para comparar el impacto sobre la tasa de acierto del sistema en una tarea de clasificación de utilizar una distribución de umbrales homógenea y heterogénea en las KCs. La distribución de umbrales heterogénea se ajusta para cada conjunto de datos con un algoritmo que encuentra la mejor distribución con una técnica de fuerza bruta. Los resultados muestran que la capacidad de discriminación del sistema mejora sustancialmente cuando se utiliza una distribución de umbrales heterogénea [5, 6]. Además, estos umbrales se mantendrían en el intervalo $[0 - 15]$, lo que estaría de acuerdo con los resultados obtenidos mediante los modelos probabilísticos [8, 41]. También se muestra que, dependiendo de la complejidad de los patrones que el sistema tenga que clasificar, se requieren de diferentes balances entre neuronas especialistas y generalistas, balance que está determinado por la distribución heterogénea de umbrales neuronales [7].

Este trabajo toma como partida el modelo anterior para desarrollar otro diferente en el que se incluya un algoritmo de aprendizaje capaz de ajustar la distribución de umbrales óptima en las KCs para cada problema de clasificación presentado en la red. Además, también se introducirán los efectos de la inhibición procedente de la GGN sobre esta capa para limitar su nivel de actividad por debajo de cierto límite y poder controlar el nivel de inyectividad del código disperso con el que el sistema está trabajando y sus efectos sobre la excitabilidad de las KCs.

3

Diseño y desarrollo

En el siguiente capítulo se detalla el modelo de sistema olfativo de la langosta que se ha desarrollado, las diferentes estrategias que incorpora para el reconocimiento de patrones y sus parámetros. También se presenta el algoritmo de aprendizaje empleado por el modelo y los problemas de clasificación utilizados para evaluar su capacidad de discriminación en diferentes modos de funcionamiento.

3.1. Características generales del modelo del sistema olfativo de la langosta voladora

El modelo propuesto del sistema olfativo de la langosta se basa en una red neuronal de una sola capa oculta con aprendizaje supervisado que incorpora diferentes estrategias empleadas en el sistema biológico para la discriminación de odorantes. Dentro de estas estrategias destaca el ajuste de la excitabilidad de las KCs por medio de sus umbrales de disparo para regular la actividad en esta capa y favorecer la representación dispersa de los estímulos.

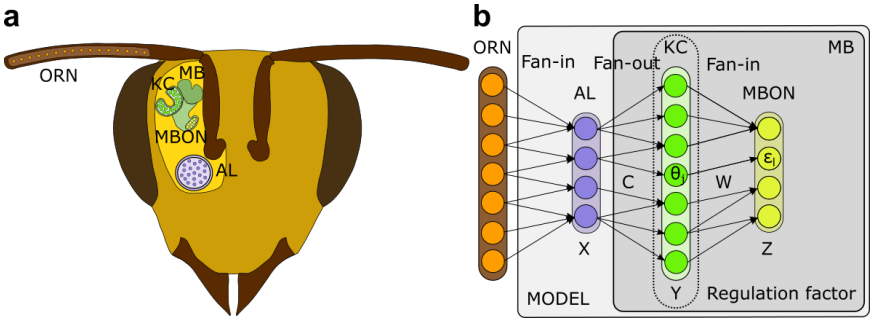

Figura 3.1: Modelo del sistema olfativo del insecto. **a** Estructura del sistema biológico **b** Modelo computacional del sistema olfativo basado en una red neuronal de una sola capa oculta

La red modela tanto el lóbulo antenal como el cuerpo fungiforme y consta de tres capas, como puede observarse en la Figura 3.1 Panel b. La capa de entrada X se corresponde con las neuronas PNs que proyectan su actividad mediante la matriz de conectividad aleatoria C a

la capa oculta Y. La capa Y se corresponde con las KCs, que tienen un umbral de disparo θ. Por último, la capa de salida Z se corresponde con las MBONs, cuyo umbral de disparo es ϵ y que están conectadas a las KCs mediante la matriz de conectividad W, donde tiene lugar el aprendizaje.

A continuación se explican las características y las estrategias de inspiración biológica que incluye este modelo:

- **Mecanismo de control de ganancia:** los patrones que llegan a la capa de entrada X, correspondiente a las PNs, se transforman mediante un mecanismo sencillo de control de ganancia. El objetivo es lograr una representación de los patrones independiente de su intensidad, considerando la intensidad como el nivel de activación que provocan en las neuronas PNs, de forma que se facilite su identificación cuando se proyecte a las capas siguientes de la red. El control de ganancia incluido en el modelo simplemente hace una re-normalización de los patrones de forma que la activación de las neuronas PNs sea uniforme para todos los patrones, con independencia de su intensidad.

- **Conectividad entre PNs y KCs:** para modelar las conexiones entre las PNs (capa X) y las KCs (capa Y), seguimos el método que más se ha utilizado en los modelos del sistema olfativo de los insectos y que consiste en utilizar una matriz aleatoria de conexiones binarias, C, donde cada posible conexión entre una PN y una KC existe con probabilidad p_c. La probabilidad de conexión p_c es un parámetro del modelo que puede tomar diferentes valores desde $p_c = 0{,}0$, valor para el que no existiría ninguna conexión, hasta $p_c = 1{,}0$, donde todas las PNs estarían conectadas a todas las KCs.

  El motivo por el que se emplea una matriz estocástica C para modelar estas conexiones se debe a que no existe reproducibilidad en la conectividad PNs-KCs entre individuos de la misma especie y tampoco existe aprendizaje en esta capa, como se ha comentado en la Sección 2.4, por lo que se demuestra que una matriz estocástica es suficiente para modelarlas [8, 41].

- **Divergencia en la proyección de las PNs hacia las KCs:** uno de los parámetros clave para conseguir la codificación dispersa en la capa de las KCs es el tamaño de la red, como se muestra en [41, 31]. Esto es así porque, para conseguir que los estímulos se representen en un espacio de mayor dimensionalidad de tal forma que sólo subconjuntos pequeños de neuronas respondan ante diferentes estímulos, es necesario que el número de KCs sea muy superior al de PNs. En la langosta, se estima que la cantidad de PNs en la red sería de 830 neuronas, mientras las KCs serían 50 000 [55]. Con estos datos, el ratio PN-KC en la red es de 1:50. En el modelo presentado este ratio se mantiene para asegurar la proyección divergente de los patrones de clasificación en la capa de las KCs y facilitar la codificación dispersa.

- **Conectividad entre KCs y MBONs:** la conectividad entre la capa Y y la capa Z es de todos con todos y se representa mediante la matriz de pesos W. Como en esta capa se lleva a cabo el aprendizaje asociativo entre los patrones y las salidas de la red [32], los pesos de W se ajustan mediante el algoritmo de aprendizaje de retropropagación que se explica en la Sección 3.2.4.

- **Inhibición mutua en las MBONs:** el número de MBONs en la capa Z se corresponde con el número de clases en el problema de clasificación. Para elegir a qué clase pertenece cada patrón, se elige la de la neurona que muestra una mayor activación, siguiendo la estrategia *winner-takes-all*. Esta estrategia tendría el mismo efecto que la inhibición mutua entre estas neuronas y, junto al aprendizaje asociativo, es un mecanismo habitual para simular el aprendizaje en las redes biológicas.

- **Ajuste de los umbrales de disparo de las KCs y MBONs:** el aprendizaje de los umbrales θ de las KCs y ϵ de las MBONs se realiza mediante el algoritmo de retropropagación según se detalla en la Sección 3.2.4.

- **Regulación de la actividad en la capa de las KCs:** el modelo incluye un mecanismo para regular la actividad de las KCs a largo plazo a través de su umbral neuronal. Este mecanismo representaría los efectos de la inhibición que las KCs reciben por parte de la GGN en función del nivel de activación que alcancen. Los detalles del mecanismo de regulación se presentan en la Sección 3.2.3.

3.2. Algoritmo de aprendizaje del modelo

En esta sección se presentan los detalles del algoritmo de aprendizaje supervisado utilizado por el modelo de red neuronal introducido en la Sección 3.1. Del mismo modo, se describe la derivación de las reglas de actualización para los pesos W entre la capa oculta Y y la salida Z, los umbrales θ de las KCs y los umbrales ϵ de las MBONs, así como el mecanismo de regulación empleado para controlar el nivel de activación en las KCs y el código disperso.

3.2.1. Activación de las neuronas

En esta sección se describen las funciones de activación que se han elegido para las neuronas de la red. En el caso de las neuronas PNs de la capa de entrada, su activación se corresponde directamente con los patrones de entrada que llegan a la red tras ser preprocesados mediante un mecanismo de control de ganancia, por lo que la función de activación en este caso es la función identidad aplicada a cada patrón.

En el caso de las KCs, la mejor forma de modelarlas es mediante el modelo de McCulloch-Pitts, ya que estas neuronas funcionan como detectores de coincidencias que sólo disparan si su entrada supera cierto umbral. Sin embargo, al utilizar el algoritmo de retropropagación para el aprendizaje en la red, es necesario que la función de activación sea derivable para poder calcular el gradiente de la función de coste y las reglas para modificar los parámetros de la red. Por tanto, la función Heaviside empleada en el modelo de McCulloch-Pitts no es adecuada y se ha sustituido por una función sigmoidal $\sigma(x)$ en el rango [0.0,1.0]. Por tanto, la expresión de la activación u_j de la neurona y_j de la capa oculta sería:

$$u_j = \sigma(y_j) = \sigma\left(\sum_{i=1}^{N_{PN}} x_i c_{ij} - \theta_j\right) , \tag{3.1}$$

donde x_i es el valor de activación de las PN, c_{ij} es la conexión entre la PN x_i y la neurona KC y_j y puede tomar los valores 0 o 1 dependiendo de si existe una conexión entre las diferentes neuronas y θ_j es el umbral de disparo de la neurona y_j.

Las neuronas MBONs de la capa de salida Z también se modelan como en el caso anterior, por lo que la expresión para su nivel de activación, a_k de la neurona de la capa de salida z_k sería:

$$a_k = \sigma(z_k) = \sigma\left(\sum_{j=1}^{N_{KC}} u_j w_{jk} - \epsilon_k\right) , \tag{3.2}$$

donde u_j es el valor de activación de la KC, w_{jk} es el peso de la conexión entre la KC u_j y la MBON z_k y ϵ_i es el umbral de disparo de la neurona z_j.

3.2.2. Selección de la función de coste

El algoritmo de retropropagación que se emplea para ajustar los pesos W, θ y ϵ del modelo se basa en el método matemático de descenso por gradiente, el cual minimiza una función de coste que mide el error entre las predicciones hechas por la red y la clase real de los patrones de entrenamiento. La elección de esta función de coste es determinante a la hora de calcular las actualizaciones de los parámetros de la red durante la fase de entrenamiento y tiene una gran influencia en el proceso de convergencia del modelo. Por tanto, la elección de una función de coste adecuada es esencial para que el modelo funcione correctamente.

Una de las funciones de coste que se usan más comúnmente en aprendizaje supervisado es el error cuadrático medio, que se define como:

$$MSE(y,t) = \frac{1}{2}\frac{1}{N}\sum_{i=0}^{N}(y_i - t_i)^2 \ , \tag{3.3}$$

donde N es el número total de clases, y es el vector de predicciones para un patrón y t es el vector de clases objetivo.

Para poder aplicar el algoritmo de retropropagación para ajustar los pesos W de la red con el error cuadrático medio como función de coste, es necesario calcular su derivada en función de los pesos. La actualización del peso w_{jk} que conecta la neurona z_j de la capa de salida con la neurona y_k de la capa oculta sería:

$$\frac{\delta E}{\delta w_{jk}} = \frac{\delta E}{\delta a_k}\frac{\delta a_k}{\delta z_k}\frac{\delta z_k}{w_{jk}} = (a_k - t_k)\sigma'(z_k)u_j \tag{3.4}$$

En el caso de los umbrales neuronales de las MBONs (ϵ) y las KCs (θ), su actualización utilizando el MSE sería:

$$\frac{\delta E}{\delta \epsilon_k} = \frac{\delta E}{\delta a_k}\frac{\delta a_k}{\delta z_k}\frac{\delta z_k}{\delta \epsilon_k} = -(a_k - t_k)\sigma'(z_k) \tag{3.5}$$

$$\frac{\delta E}{\delta \theta_j} = \sum_{k=1}^{N_{MBON}}\left(\frac{\delta E}{\delta a_k}\frac{\delta a_k}{\delta z_k}\frac{\delta z_k}{\delta u_j}\right)\frac{\delta u_j}{\delta y_j}\frac{\delta y_j}{\delta \theta_j} = -\sum_{k=1}^{N_{MBON}}\left((a_k - t_k)\sigma'(z_k)w_{jk}\right)\sigma'(y_j) \tag{3.6}$$

Como puede verse, todas las ecuaciones contienen el término $\sigma'(z_k)$ que puede ralentizar el aprendizaje y provocar que la red no converja a un resultado correcto. Esto se debe a que la pendiente de la función sigmoidal que se utiliza como activación en las neuronas tiende a 0 según la función se aproxima a 0 o a 1, por lo que el valor de $\sigma'(z_k)$ se irá haciendo más pequeño a medida que la red evoluciona hacia un resultado, por lo que el valor de actualización de los pesos será muy pequeño y el aprendizaje muy lento. Este fenómeno se incrementa para redes que poseen muchas neuronas en la capa oculta, como es el caso de la red neuronal que estamos modelando, en la que la proporción entre las neuronas de entrada y las de la capa oculta es 1:50. Esto hace que al comienzo del aprendizaje haya más cantidad de neuronas con valores de activación cercanos a 1 o a 0, lo que hará que su derivada, $\sigma'(y_j)$ también sea próxima a 0, lo que en combinación con lo anterior, hace que la red experimente serios problemas de convergencia. Por tanto, el error cuadrático medio es una mala elección de función de coste en este caso.

Teniendo lo anterior en cuenta, la función de coste que se ha elegido para evitar el problema de una convergencia lenta o hacia un resultado incorrecto es la entropía cruzada o *log loss* [66], que se calcula como:

$$H(y,t) = -\sum_{i=1}^{N}[t_i log(y_i) + (1 - t_i)log(1 - y_i)] \ , \tag{3.7}$$

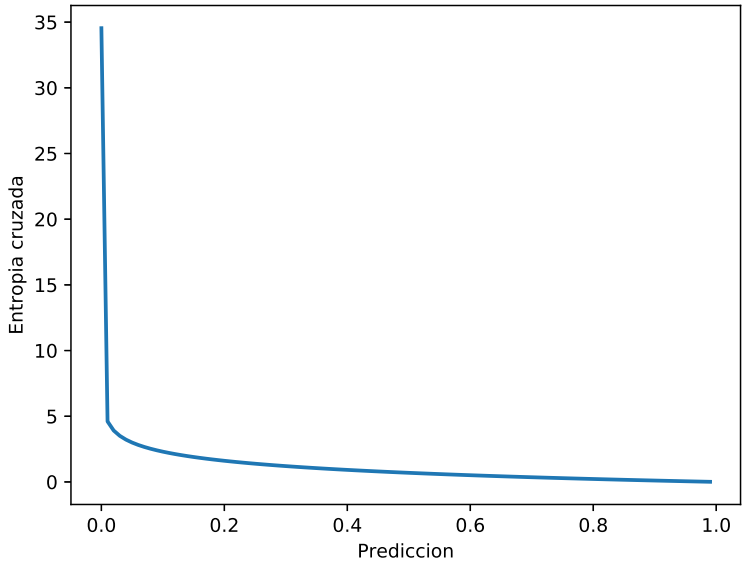

Figura 3.2: Función de entropía cruzada para diferentes valores de predicción teniendo en cuenta que el valor real de la clase es 1.

donde N es el número total de clases, y es el vector de predicciones para un patrón y t es el vector de clases objetivo.

Como puede observarse en la Figura 3.2, cuando el valor predicho se acerca a la clase, el valor de la función de entropía cruzada decrece más lentamente, mientras que si se aleja del valor correcto, crece rápidamente. De esta forma, aunque se penalicen los dos tipos de errores, el coste es mucho mayor si la predicción está más alejada del valor correcto, de forma que aumenta la velocidad de aprendizaje de la red. Sin embargo, la principal ventaja de emplear la función de entropía cruzada es que al calcular las reglas de actualización de los parámetros, la derivada de esta función posee propiedades que solucionan el problema de la ralentización de la convergencia del modelo. La derivación de las reglas de actualización de los parámetros para el modelo utilizando la entropía cruzada como función de coste se detallan en la Sección 3.2.4.

3.2.3. Regulación de la actividad de las KCs

Según lo expuesto en la Sección 2.4, el *feedback* de la neurona GGN a toda la población de KCs contribuye en gran medida a regular el nivel de actividad de las neuronas de esta capa y propicia la utilización del código disperso para codificar la información, además de asegurar la eficiencia energética. Debido a que se cree que la GGN recibe como entrada el nivel de actividad de todas las KCs y responde con una inhibición uniforme y proporcional, en el modelo se introduce un término de regulación de la actividad de las KCs que imita este comportamiento. El término de regulación de la actividad (ART, de sus siglas en inglés *Activity Regulation Term*) es el siguiente:

$$ART(U) = \frac{1}{2}\left(\frac{1}{N_{KC}}\sum_{i=1}^{N_{KC}} u_i - s\right)^2 , \tag{3.8}$$

donde N_{KC} es el número de KCs en la capa oculta, U es el vector de activaciones de todas las KCs y $s \in [0{,}0, 1{,}0]$ es el parámetro que permite controlar el nivel de actividad en la capa de KCs desde ninguna actividad con $s = 0{,}0$ hasta la actividad máxima con $s = 1{,}0$. Como puede verse,

el término de regulación calcula la media de actividad en la capa de las KCs y la compara con el nivel de actividad s que se quiere mantener, de forma que la excitabilidad de estas neuronas se regule en consecuencia mediante su umbral.

3.2.4. Derivación de reglas de aprendizaje

Teniendo en cuenta lo expuesto en las secciones 3.2.2 y 3.2.3, la función a minimizar por la red neuronal del modelo mediante el ajuste de sus pesos W y los umbrales θ y ϵ para resolver el problema de clasificación es:

$$E(A, T, U) = H(A, T) + ART(U) \ , \tag{3.9}$$

donde T es el vector de clases objetivo, A es la activación de las MBONs de la capa de salida, U es la activación de las KCs de la capa oculta, $H(x)$ la entropía cruzada y $ART(x)$ el término de regulación de la actividad de las KCs.

A partir de la expresión para $E(A, T, U)$, calculamos la regla de actualización de los pesos de la matriz W:

$$\frac{\delta H}{\delta w_{jk}} = \frac{\delta H}{\delta a_k}\frac{\delta a_k}{\delta z_k}\frac{\delta z_k}{\delta w_{jk}} = -\left(\frac{t_k}{\sigma(z_k)} - \frac{(1-t_k)}{1-\sigma(z_k)}\right)\sigma'(z_k)y_j =$$

$$= -\frac{(1-\sigma(z_k))t_j - \sigma(z_k)(1-t_k)}{\sigma(z_k)(1-\sigma(z_k))}\sigma'(z_k)y_j = \frac{\sigma(z_k) - t_k}{\sigma(z_k)(1-\sigma(z_k))}\sigma'(z_k)y_j$$

Teniendo en cuenta que $\sigma'(z_k) = \sigma(z_k)(1-\sigma(z_k))$, se obtiene:

$$\frac{(1-\sigma(z_k))t_k - \sigma(z_k)(1-t_k)}{\sigma(z_k)(1-\sigma(z_k))}\sigma'(z_k)y_j =$$

$$= \frac{(1-\sigma(z_k))t_k - \sigma(z_k)(1-t_k)}{\sigma(z_k)(1-\sigma(z_k))}\sigma(z_k)(1-\sigma(z_k))y_j =$$

$$= (\sigma(z_k) - t_k)y_j$$

Por tanto, la regla de actualización para los pesos de la matriz W será:

$$w_{jk_{t+1}} = w_{jk_t} + \alpha\frac{\delta H}{\delta w_{jk}} = w_t + \alpha(\sigma(z_k) - t_k)y_j \tag{3.10}$$

donde α es la tasa de aprendizaje de la red y t es la época de aprendizaje.

Como puede verse, esta regla de actualización no incluye el término $\sigma'(z_k)$ responsable de los problemas de convergencia de la red que se expusieron en la Sección 3.2.2, de forma que la actualización de los parámetros es solamente proporcional al error cometido por la red en su predicción de las clases, sin que intervengan otros factores.

La regla de actualización para los umbrales ϵ de las MBONs se deriva de una forma similar:

$$\frac{\delta H}{\delta \epsilon_k} = \frac{\delta H}{\delta a_k}\frac{\delta a_k}{\delta z_k}\frac{\delta z_k}{\delta \epsilon_k} = -(\sigma(z_k) - t_k)$$

$$\epsilon_{k_{t+1}} = \epsilon_{k_t} + \alpha\frac{\delta H}{\delta \epsilon_k} = \epsilon_{k_t} - \alpha(\sigma(z_k) - t_k) \tag{3.11}$$

Por último, en la regla de actualización para los umbrales θ de las KCs se deriva de la siguiente forma:

$$\frac{\delta E}{\delta \theta_j} = \sum_{k=1}^{N_{MBON}}\left(\frac{\delta H}{\delta a_k}\frac{\delta a_k}{\delta z_k}\frac{\delta z_k}{\delta u_j}\right)\frac{\delta u_j}{\delta y_j}\frac{\delta y_j}{\delta \theta_j} + \frac{\delta ART}{\delta u_j}\frac{\delta u_j}{\delta y_j}\frac{\delta y_j}{\delta \theta_j} =$$

$$= -\sum_{k=1}^{N_{MBON}} ((\sigma(z_k) - t_k)w_{jk})\,\sigma'(y_j) - \left(\frac{1}{N_{KC}}\sum_{i=1}^{N_{KC}} u_j - s\right)\sigma'(y_j) =$$

$$= -\sigma'(y_j)\left(\sum_{k=1}^{N_{MBON}} ((\sigma(z_k - t_k))w_{jk}) + \frac{1}{N_{KC}}\sum_{i=1}^{N_{KC}} u_j - s\right)$$

$$\theta_{j_{t+1}} = \theta_{j_t} + \alpha\frac{\delta E}{\delta\theta_j} = \theta_{j_t} - \alpha\sigma'(y_j)\left(\sum_{k=1}^{N_{MBON}} ((\sigma(z_k - t_k))w_{jk}) + \frac{1}{N_{KC}}\sum_{i=1}^{N_{KC}} u_j - s\right) \quad (3.12)$$

En esta regla de actualización para los umbrales θ es donde tiene influencia el término de regulación de actividad (ART), de forma que la excitabilidad de las neuronas de esta capa se mantenga por debajo de cierto límite dado por el parámetro s, con el fin de controlar el gasto energético y el grado de inyectividad del código disperso utilizado.

3.2.5. Optimización de la red

Con el objetivo de evitar problemas para alcanzar la solución correcta y acelerar la convergencia del modelo, se introducen las técnicas de optimización que se describen a continuación:

- **Descenso por gradiente con *minibatch*:** en lugar de realizar las actualizaciones de pesos y umbrales por cada patrón (*online*) o una vez por época cuando se han mostrado a la red todos los patrones de entrenamiento (*batch*), se define un tamaño de *batch* determinado y los patrones de entrenamiento se dividen en grupos de ese tamaño, de modo que las actualizaciones de pesos se realizan varias veces por época, cada vez que se muestra a la red un grupo completo de patrones. Esta alternativa tiene la ventaja de que evita que el sistema quede atrapado en mínimos locales de la función de coste o que el sistema se pase la solución óptima [67].

- ***Momentum***: el objetivo de esta técnica es acelerar la convergencia y a la vez minimizar las oscilaciones en el proceso de descenso por gradiente. Para implementarlo, basta con añadir a la actualización de parámetros un nuevo término que consiste en el valor de actualización de parámetros del paso anterior multiplicado por un factor μ. Por ejemplo, la expresión para la actualización de los pesos W incorporando *momentum* sería:

  $$w_{jk_{t+1}} = w_{jk_t} + \alpha\frac{\delta H}{\delta w_{jk}}_t + \mu\left(\alpha\frac{\delta H}{\delta w_{jk}}\right)_{t-1} \quad (3.13)$$

  Para la actualización de los umbrales θ y ϵ se aplicaría de la misma forma. La selección del valor del parámetro μ se realiza mediante prueba y error. Si se elige un valor de μ alto, la tasa de aprendizaje α debe ser un valor pequeño para asegurar la convergencia y al revés [67].

  Esta técnica ayuda a la convergencia de la red ya que acelera el aprendizaje durante las primeras épocas, previniendo que la red quede atrapada en mínimos locales o que se produzcan oscilaciones, a la vez que evita que el sistema se pase la solución óptima ya que a medida que las actualizaciones de los pesos se reducen al irse aproximando a esta solución, el valor de *momentum* se reduce también. De esta forma, actúa como un tasa de aprendizaje adaptativa e individual para cada peso.

El modelo no emplea otras técnicas muy utilizadas en redes neuronales como la regularización de los pesos para prevenir el sobreajuste a los patrones de entrenamiento debido a que resulta difícil justificar su inclusión desde un punto de vista biológico.

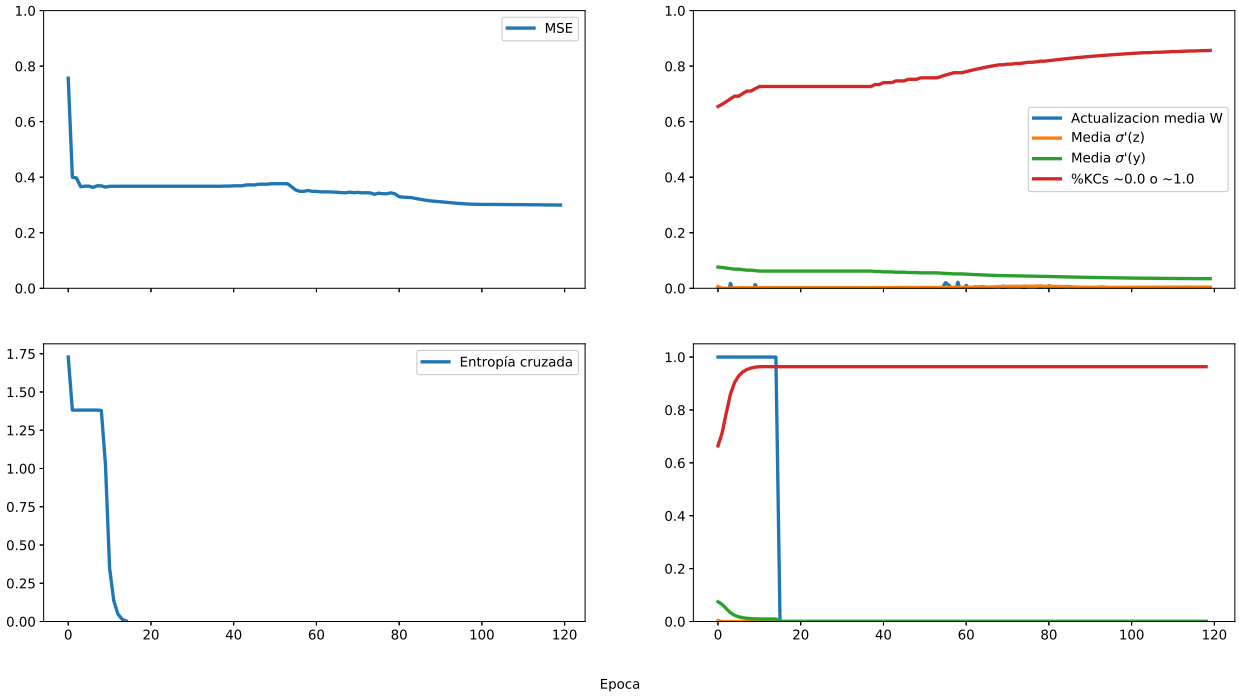

Figura 3.3: Comparativa entre el uso del MSE (fila superior) como función de coste y la entropía cruzada (fila inferior). En la parte derecha de la figura se muestra la evolución del MSE y de la entropía cruzada en función del número de épocas, mientras en la parte izquierda se muestran la evolución de diferentes parámetros y medidas del sisema. Entre estas medidas destacan el valor de actualización de los pesos en color azul y una medida de la convergencia de la red en rojo que se corresponde con el porcentaje de KCs de la capa oculta que poseen valores medios de activación cercanos a 0 o a 1.

3.2.6. Comparativa entropía cruzada vs. MSE

El objetivo de este apartado es ilustrar lo que se explicó en la Sección 3.2.2 y la Sección 3.3.3 sobre la convergencia del modelo con diferentes funciones de coste. Para ello, se ha realizado una simulación del modelo en la que tenía que resolver un problema de clasificación de dificultad baja con patrones de 250 atributos y 10 clases diferentes. Las características del problema se explican en detalle en la Sección 3.3.1 y se puede observar una muestra de los patrones en la Figura 3.4 fila superior centro. En una de las simulaciones se utiliza el error cuadrático medio (MSE) como función de coste, mientras en la otra se emplea la entropía cruzada. El resto de parámetros se mantienen igual en ambas simulaciones y no se incluye el término de regulación de la actividad de las KCs (ART), aunque sí se incorporan en ambos las optimizaciones de la Sección 3.2.5. La comparativa se presenta en la Figura 3.3.

En el caso del uso de MSE, éste solo es capaz de descender hasta un valor de 0.3 en 120 épocas, lo que se corresponde con un error de clasificación del 60 %. Mientras, la entropía cruzada consigue descender hasta cero, lo que se corresponde con un error de clasificación del 0 %, en tan solo 20 épocas. Por tanto, el uso de esta función de coste presenta una gran ventaja respecto al MSE. La razón de esto se debe, como se apuntó con anterioridad, a que en las reglas de aprendizaje para MSE (Ecuaciones 3.4, 3.5 y 3.6), se incluye el término $\sigma'(z)$. Como puede verse en el gráfico de la fila superior derecha, $\sigma'(z)$ tiene un valor muy cercano a cero durante todas las épocas de entrenamiento debido a que las neuronas de la capa de salida tienen un valor

de activación cercano a 1.0 o a 0.0. Además, al comienzo del aprendizaje, más del 60 % de las neuronas KCs de la capa oculta posee valores de activación cuya derivada es cercana a 0.0. Como consecuencia de esto, el valor de las actualizaciones de los parámetros de la red también son muy cercanas a cero. En el gráfico sólo se ha incluido el valor para la actualización de los pesos de la matriz W, pero en el caso de los parámetros θ y ϵ que también incluyen el término $\sigma'(z)$ además de $\sigma'(y)$ (Ecuación 3.6), cuyo valor también es siempre próximo a 0, ocurre el mismo problema. Si medimos el porcentaje de neuronas KCs de la capa oculta que presentan valores de activación cercanos a 0.0 o a 1.0 como medida de la convergencia de la red, podemos ver que apenas llega al 80 % tras 120 épocas.

En el caso de la entropía cruzada, puede verse cómo los valores de actualización de los parámetros del modelo son lo suficientemente grandes como para modificar su comportamiento. Aunque los valores de las derivadas de la función de activación sigan siendo cercanas a cero, como no se incluyen en las reglas de aprendizaje derivadas a partir de la entropía cruzada, no tienen efecto en la convergencia de la red. Como puede verse, la cantidad de KCs con valores cercanos a 0.0 o 1.0 sube rápidamente hasta el 95 %, lo que indica que el modelo ha convergido satisfactoriamente.

3.3. Conjuntos de datos

En esta sección se presentan los diferentes problemas de clasificación con los que se comprobará la capacidad de discriminación del modelo de sistema olfativo de la langosta para diferentes configuraciones de sus parámetros.

3.3.1. Patrones gaussianos

En [7] se introduce un modelo para representar los estímulos olfativos que llegan a las PNs mediante patrones gaussianos teniendo en cuenta que en estas neuronas se hace una codificación espacial de la información además de temporal. En este modelo de representación, el nivel de actividad que un olor produce en las neuronas PNs se distribuye según una normal con cierto nivel de ruido. Las similitudes entre diferentes odorantes y su concentración dan lugar a solapamientos en la representación de odorantes distintos o a representaciones divergentes del mismo odorante, lo que aumenta la complejidad del problema y hace más difícil la correcta identificación de los patrones pertenecientes a la misma clase. A partir del grado de solape intra e inter clase, se desarrolla una medida de complejidad para poder aplicarla a diferentes conjuntos de datos y compararlos entre ellos. El solape de los patrones inter-clase se controla mediante la desviación estándar de las gaussianas y la variación intra-clase mediante la cantidad de ruido que se añade a los patrones.

En este trabajo se van a emplear diferentes conjuntos de estos patrones gaussianos con diferentes niveles de complejidad, según el solape entre diferentes clases y la cantidad de ruido, para estudiar la capacidad de discriminación del modelo de sistema olfativo desarrollado y la influencia del umbral neuronal en la misma. En concreto, se emplearán seis conjuntos de datos con 1000 patrones cada uno. Algunos ejemplos de los diferentes conjuntos de gaussianas utilizados se pueden observar en la Figura 3.4 junto con los valores de desviación típica y ruido correspondientes a cada conjunto. Los patrones constan de 250 atributos que representan el nivel de activación de cada PN, y de diez clases. Más detalles sobre la medida de complejidad y la generación de los patrones se pueden consultar en [7].

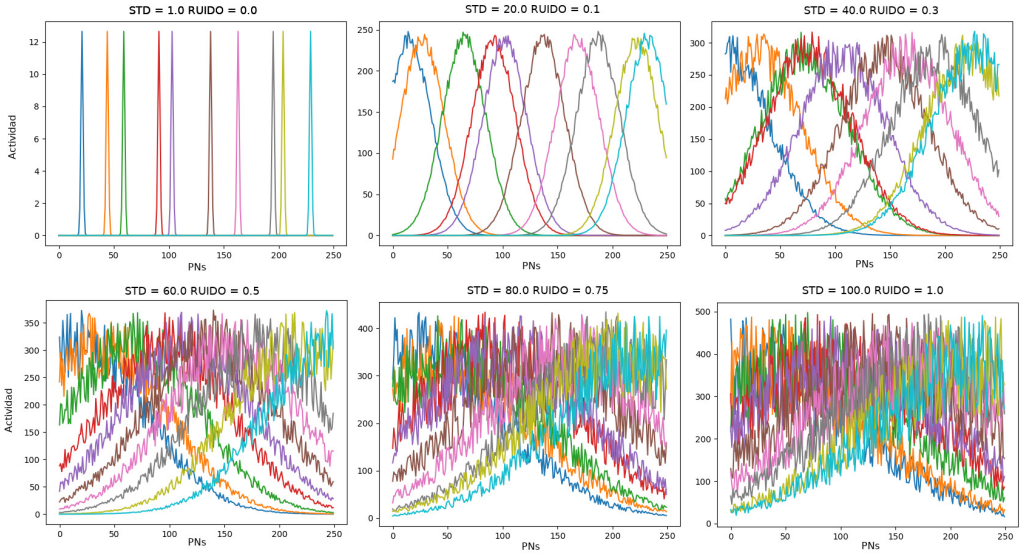

Figura 3.4: Patrones gaussianos de complejidad creciente de izquierda a derecha y de arriba a abajo, según el solapamiento de las clases (debido a la desviación estándar de las gaussians, STD) y el nivel de ruido. El eje X se corresponde con las 250 neuronas PNs y el eje Y con la activación de cada neurona. En cada uno de los gráficos se muestra un ejemplo de patrón por cada una de las 10 clases existentes.

3.3.2. Patrones MNIST

El conjunto de patrones MNIST es un problema de clasificación muy conocido y utilizado en aprendizaje automático, por lo que se tiene conocimiento sobre los errores de clasificación que deberían obtenerse. Además, este conjunto de datos también se ha empleado para evaluar otros modelos del sistema olfativo basados en redes neuronales [6, 7], de forma que permite establecer comparaciones entre ellos. El error de clasificación para MNIST se situaría entre el 0 % y el 20 % para diferentes métodos de clasificación como SVMs, perceptrón multicapa o KNNs [7].

Los patrones MNIST están formados por dígitos manuscritos representados mediante imágenes de 28x28 píxeles y 10 posibles clases. Para poder ser presentados a la red neuronal, las imágenes se representan en un vector de una sola dimensión con 784 atributos. En la Figura 3.5 se puede observar la representación de 10 patrones, cada uno perteneciente a una de las posibles clases.

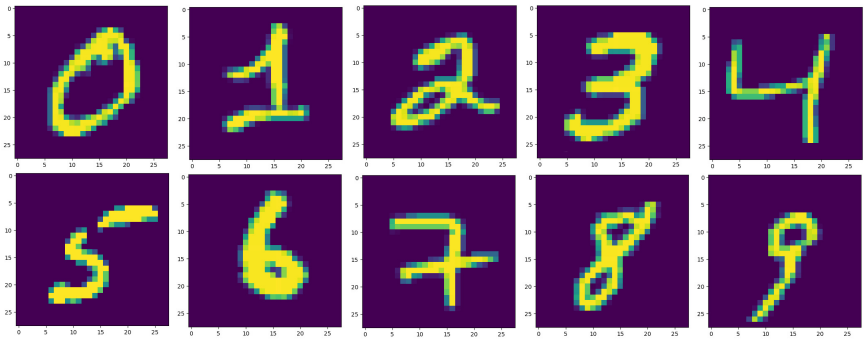

Figura 3.5: Selección de patrones extraídos de la base de datos MNIST.

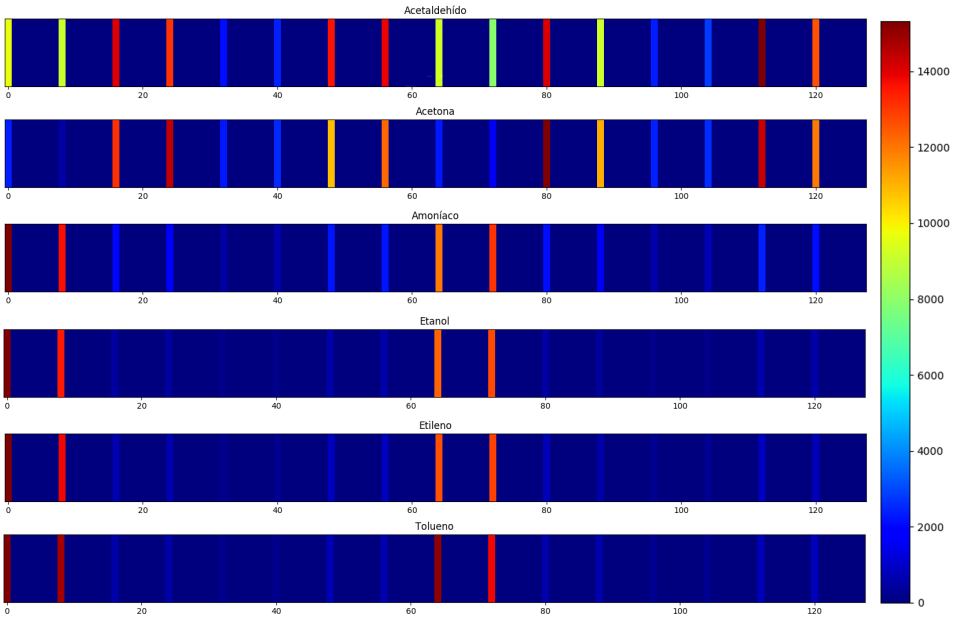

Figura 3.6: Vectores de 128 características correspondientes a cada uno de los seis posibles odorantes a una concentración de 50ppmv. El conjunto de datos han sido obtenidos de [68]

3.3.3. Odorantes captados por narices electrónicas

Para presentar al modelo patrones parecidos a los que el sistema biológico tendría que identificar en la naturaleza, utilizamos una base de datos de odorantes captados por narices electrónicas publicada en [68]. La información para cada presentación de un odorante se capturó mediante un array de 8 sensores y se extrayeron 16 características de la señal de cada sensor, por lo que el número de atributos totales es de 128. Los posibles odorantes son 6: acetaldehído, acetona, amoníaco, etanol, etileno y tolueno. Los odorantes se presentan a diferentes concentraciones en cada prueba. En la Figura 3.6 se puede observar una representación gráfica de un patrón correspondiente a cada uno de los odorantes anteriores.

La base de datos contiene patrones correspondientes a tres años de mediciones divididos en 10 conjuntos de datos. Durante el tiempo que duraron los experimentos, los sensores sufrieron desgaste y contaminación, por lo que la tarea de clasificación de los patrones es complicada debido a la variabilidad que introducen estos factores. En [68] se presentan los resultados de clasificación de estos datos mediante SVMs, por lo que se pueden establecer comparaciones con la tasa de acierto obtenida por el modelo del sistema olfativo que se ha propuesto.

4

Experimentos y análisis de resultados

En este capítulo se muestran los resultados obtenidos tras evaluar el modelo del sistema olfativo de la langosta con los conjuntos de datos propuestos en el Capítulo 3 y diferentes combinaciones de sus parámetros.

4.1. Realización de las simulaciones y parámetros

El modelo se ha evaluado mediante los tres conjuntos de datos propuestos en la Sección 3.3 y diferentes combinaciones del nivel de actividad en la capa de las KCs del modelo (parámetro s) y la probabilidad de conexión entre las neuronas PNs y KCs (parámetro p_c).

Los valores de p_c que se han elegido para las simulaciones son $0{,}01, 0{,}1, 0{,}2, 0{,}3, 0{,}5$. Teniendo esto en cuenta, los valores de s se han seleccionado de tal forma que pueda comprobarse el funcionamiento del modelo para actividad muy baja, baja, media y alta. La elección de estos valores se justifica en que están dentro del rango que se ha propuesto como biológicamente posible [9, 49, 50, 55, 41]. En la Tabla 4.1 pueden observarse los valores de todos los parámetros del sistema que se han elegido para cada conjunto de datos.

Todos los resultados que se muestran en esta sección se han obtenido promediando 10 simulaciones para cada combinación de parámetros p_c y s. En los casos de los patrones gaussianos y los patrones de MNIST se ha utilizado validación cruzada con 5 iteraciones. En el caso de los odorantes captados por narices electrónicas, debido a que la estructura de los datos experimenta cambios en función del tiempo, se ha utilizado validación simple.

4.2. Preprocesamiento de los patrones

Los atributos de los diferentes conjuntos de datos se han normalizado mediante la unidad tipificada (*Z-Score*) y además se ha aplicado el método sencillo de control de ganancia explicado en la Sección 3.1 antes de presentárselos a la red.

Debido a la excesiva dimensionalidad de la red para los patrones del MNIST, con 784 atributos y $\sim$ 40000 neuronas KCs (debido a la proporción 1:50 entre entrada y capa oculta), los

Datos	Preprocesamiento	Validación	Tamaño red	s	p_c	α	μ
Gaussianas	Normalización Z-Score	Validación cruzada 5 iteraciones	250x12501x10	0.1 (actividad 0 %-20 %) 0.5 (actividad 20 %-30 %) 0.9 (actividad 30 %-50 %)	$p_c \in [0{,}01 - 0{,}5]$	0.015	0.7
MNIST	Normalización Z-Score Reducción dimensionalidad	Validación cruzada 5 iteraciones	225x11251x10	0.05 (actividad 0 %-5 %) 0.1 (actividad 0 %-15 %) 0.5 (actividad 20 %-30 %) 0.9 (actividad 30 %-50 %)	$p_c \in [0{,}01 - 0{,}5]$	0.01 0.01 0.007 0.005	0.7
Odorantes	Normalización Z-Score	Validación simple	128x6401x6	0.1 (actividad 0 %-20 %) 0.5 (actividad 20 %-30 %) 0.9 (actividad 30 %-50 %)	$p_c \in [0{,}01 - 0{,}5]$	0.01	0.7

Cuadro 4.1: Tabla de parámetros del modelo para los diferentes conjuntos de datos

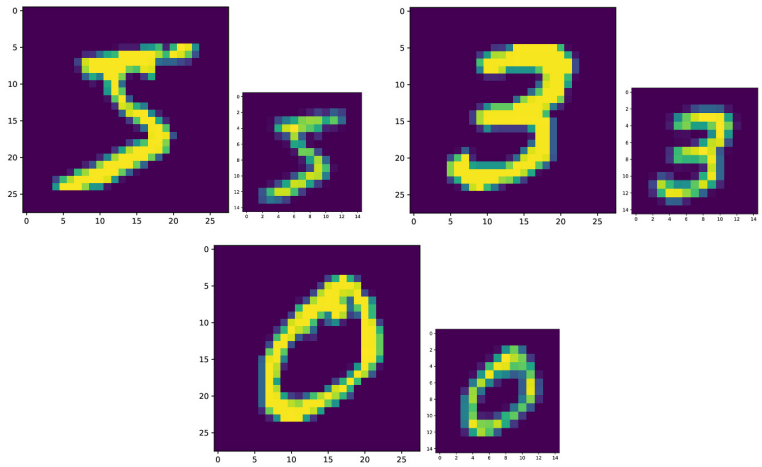

Figura 4.1: Comparativa entre los patrones MNIST antes y después de reducir su tamaño para ajustarlos a las características de la red y así facilitar y acelerar la convergencia del modelo.

patrones se han preprocesado para reducir su dimensionalidad, sin alterarlos en la medida de lo posible, de cara a garantizar la convergencia del modelo y acelerarla. Para ello, se ha modificado el tamaño de los patrones para que, en vez de ser imágenes de 28x28 píxeles, estén representados por imágenes de 15x15 píxeles. Así, cada patrón pasa a estar formado por 225 atributos y el tamaño de la capa oculta correspondiente es de $\sim$ 11000 neuronas KCs.

La transformación de los patrones MNIST se ha llevado a cabo mediante la función *resize()* de la librería cv2 de Python. Esta función escala imágenes utilizando diferentes métodos de interpolación. El método de interpolación que se ha utilizado en este caso es la interpolación de área, ya que produce mejores resultados para reducción de imágenes [69]. Este tipo de interpolación establece un tamaño de ventana determinado y calcula el valor de cada píxel de la imagen reducida como el promedio de los píxeles dentro de esa ventana en la imagen original. En la Figura 4.1 se puede observar una comparación entre los patrones antes y después de esta transformación. Como puede verse, no se altera ni la complejidad de los patrones ni su estructura.

4.3. Resultados para patrones gaussianos

En esta sección se presentan y discuten los resultados que se han obtenido tras comprobar el funcionamiento del modelo clasificando los patrones gaussianos de diferentes complejidades presentados en Sección 3.3.2. Se muestran tanto el error de clasificación y el nivel de actividad del

modelo para cada combinación de parámetros p_c y s como las características de las distribuciones de umbrales a las que el modelo ha converge.

4.3.1. Error de clasificación y nivel de actividad

En la Figura 4.2 se muestran el error de clasificación obtenido para los diferentes conjuntos de patrones gaussianos propuestos. La primera observación que se puede destacar sobre estos resultados es que, cuando la dificultad del problema de clasificación es baja, como ocurre en las gráficas de la parte superior de la figura, los errores obtenidos por los tres niveles de actividad son similares y ninguno supone una mejora clara sobre los demás. Sin embargo, a medida que la dificultad del problema crece, como en la parte inferior de la figura, comienza a haber diferencias en el error obtenido por cada nivel de actividad, siendo el nivel bajo el que mejores resultados obtiene para todos valores de p_c con los que se combina. Este comportamiento se debe a que, cuando los patrones de diferentes clases son fácilmente separables entre sí, el uso de la codificación dispersa no supone demasiadas ventajas, mientras que cuando existe un mayor solapamiento entre las clases y la dificultad para reconocer los patrones crece, el código disperso aumenta la separabilidad de los mismos y facilita la tarea. Si se compara el error obtenido por el nivel de actividad alto con el obtenido utilizando un nivel de actividad bajo, la mejora que supone el nivel bajo en el error de clasificación es significativa, variando entre el 10 % y el 20 % para los problemas más complejos de la fila inferior de la figura.

Estas ventajas que supone el uso del código disperso ya se expusieron en la Sección 2.3 y se han observado numerosas veces, tanto en modelos teóricos [41, 70] como en la biología [71, 32, 51, 31, 3], además de haberse desarrollado métodos de aprendizaje automático que realizan transformaciones similares con los patrones para clasificarlos, como las SVMs [41]. Sin embargo, el algoritmo desarrollado para este modelo permite controlar la calidad del código disperso utilizado mediante el control de la actividad de las neuronas con el parámetro s y mediante el ajuste de los umbrales neuronales. Por tanto, la distribución de umbrales neuronales en la población de neuronas KCs se revela como un elemento fundamental a la hora de conseguir una mejor codificación de la información sensorial.

En cuanto a la probabilidad de conexión p_c entre la capa de las PNs y las KCs, pueden realizarse tres observaciones a partir de los datos:

- Para el nivel de actividad bajo, el error mínimo de clasificación se alcanza siempre para el valor $p_c = 0{,}1$. El error aumenta ligeramente para otros valores mayores que 0,1, aunque esta variación no es muy significativa y puede decirse que el error se mantiene estable.
- En el caso del nivel de actividad medio, los valores de p_c altos, como 0,4 o 0,5, resultan más ventajosos. Para dichos valores, se alcanzan errores de clasificación cercanos, aunque ligeramente superiores, al error mínimo conseguido por la combinación de actividad baja y $p_c = 0{,}1$.
- Los resultados nos permiten descartar valores de p_c muy pequeños como 0,01, ya que como puede observarse, el error de clasificación máximo se produce en este punto para todos los conjuntos de datos utilizados.

La explicación a estos tres comportamientos radica en el balance entre la cantidad de información que se transmite entre las PNs y las KCs y las ventajas que supone el código disperso a la hora de representar los patrones y mejorar su separabilidad. Así, en el caso de p_c pequeños como ocurre con $p_c = 0{,}01$, la cantidad de información de las PNs que se transmite hacia las KCs no es suficiente para poder resolver el problema de clasificación, independientemente del nivel de actividad. En el caso del nivel de actividad medio, el código utilizado para representar los

patrones no favorece tanto su separabilidad, por lo que se necesita una cantidad de información transmitida entre PNs y KCs mayor para poder alcanzar errores semejantes a los obtenidos con el nivel de actividad bajo, razón por la que se obtienen los mejores resultados para p_c alto como 0,4 o 0,5. Por último, el mejor balance se produce en el caso de actividad baja, ya que las ventajas que supone el código disperso hacen que la cantidad de información que se transmite cuando $p_c = 0{,}1$ sea suficiente para alcanzar el error mínimo del sistema.

Como se ha visto, para estos patrones existen dos posibles combinaciones de parámetros para los que el sistema presenta una capacidad de discriminación similar: p_c pequeña con nivel de actividad bajo y p_c grande con nivel de actividad medio. Para poder decantarnos entre una de las dos soluciones, es importante tener en cuenta la eficiencia energética de ambas, ya que esta es una restricción importante para los sistemas biológicos. En la Figura 4.3 se puede observar el gasto energético asociado a cada combinación de parámetros medido como el nivel de activación medio en las neuronas KCs. El nivel de actividad bajo, que se corresponde con una activación media inferior al 20 %, es el que mejores resultados obtiene. Además, para valores de p_c pequeños como $p_c = 0{,}1$, para el que el modelo alcanza el error mínimo, supone un ahorro energético en comparación con el nivel de actividad medio de en torno a un 5 %. Por tanto, el modo de funcionamiento óptimo del modelo se corresponde con el nivel de actividad bajo y $p_c = 0{,}1$.

Por tanto, el hecho de que el error mínimo se alcance para el nivel de actividad bajo con un valor de $p_c = 0{,}1$ indica que el algoritmo desarrollado converge por sí mismo a un estado consistente con el del sistema biológico, ya que se sabe que la actividad en las KCs es baja al emplear código disperso y que el valor de p_c está en el intervalo $[0{,}1 - 0{,}5]$ [8, 70, 9]. Aunque el valor más aceptado actualmente para p_c es 0,5 [9, 49, 50], el hecho de que el el error del modelo para actividad baja no experimente grandes variaciones en función de este parámetro hace que cualquiera de estos valores sea considerado como posible.

4.3.2. Distribución de umbrales neuronales

En la Figura 4.4 se muestran los diagramas de cajas de la distribución de umbrales óptima para la combinación de parámetros que logra alcanzar el error mínimo con el gasto energético más bajo ($p_c = 0{,}1, s = 0{,}1$) para los diferentes conjuntos de patrones gaussianos. Las distribuciones que se muestran se corresponden con cada una de las cinco iteraciones de validación cruzada de una simulación del modelo por cada conjunto de patrones.

Como puede observarse, existe cierto nivel de variabilidad en las soluciones encontradas por el modelo, pero cabe destacar que el intervalo de valores en los que se encuentran los umbrales neuronales, $[0{,}0-17{,}5]$, es compatible con los resultados presentados en otros artículos [8, 41, 5, 6].

Otra de las características generales de estas distribuciones es que presentan una moderada asimetría tanto hacia los valores más bajos en algunos casos como a los más altos en otros, lo que indica que no se distribuyen siguiendo una normal. Esta asimetría es más fuerte en el caso de los patrones con una complejidad baja y se reduce a medida que aumenta la complejidad. También hay que destacar que, aunque la mayoría de soluciones consisten en distribuciones que se centran en el rango de umbrales $[7{,}5-10{,}0]$, se dan diferencias entre los niveles de complejidad. De esta forma, cuando la complejidad es baja, las soluciones incluyen tanto distribuciones con umbrales elevados como con umbrales bajos, sin mostrar preferencia por ninguna de las dos. Para la complejidad intermedia, las soluciones muestran preferencia por umbrales bajos. Por último, para complejidades altas desaparecen las distribuciones centradas en valores pequeños y predominan las distribuciones centradas en umbrales altos, además de aparecer más valores atípicos de umbrales inusualmente bajos o altos. Este comportamiento puede explicarse por la mayor cantidad de neuronas PNs que son excitadas cuando los patrones presentados son más complejos (Figura 3.2) y que en consecuencia aumentan la activación de las neuronas KCs, por

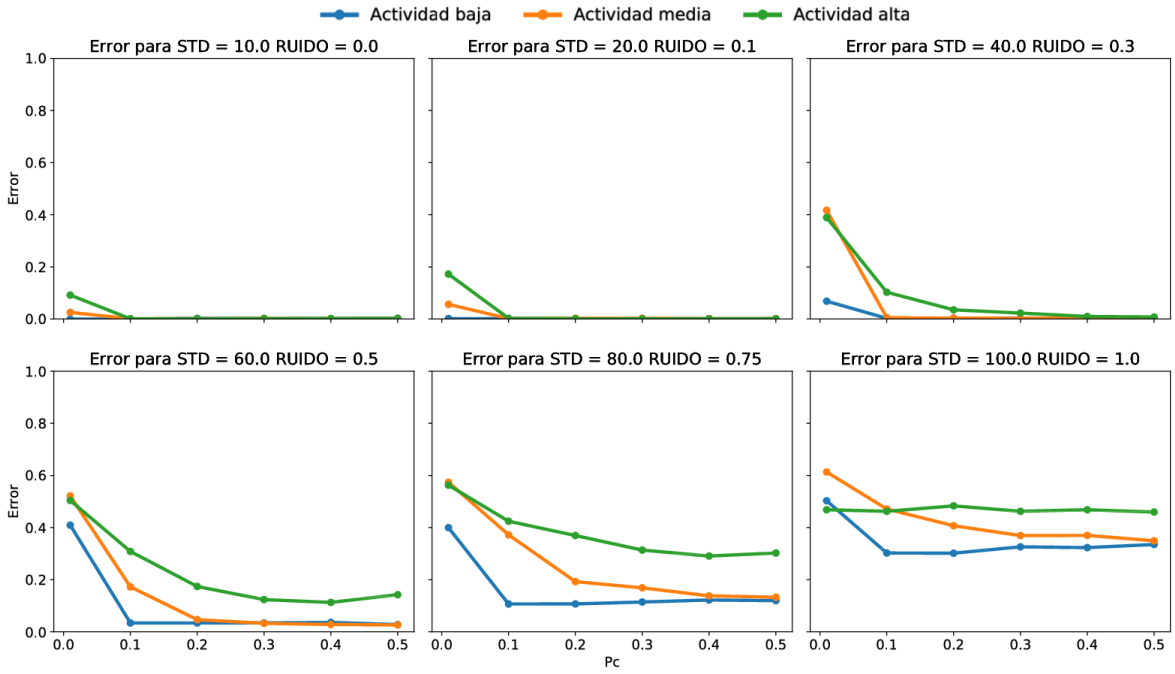

Figura 4.2: Error de clasificación obtenido por el modelo para los diferentes conjuntos de patrones gaussianos de dificultades variables en función del solapamiento entre las clases (consultar Sección 3.3.2). En cada gráfico se presenta el error para los tres niveles de actividad considerados en combinación con diferentes valores de p_c en el intervalo $[0{,}01, 0{,}5]$.

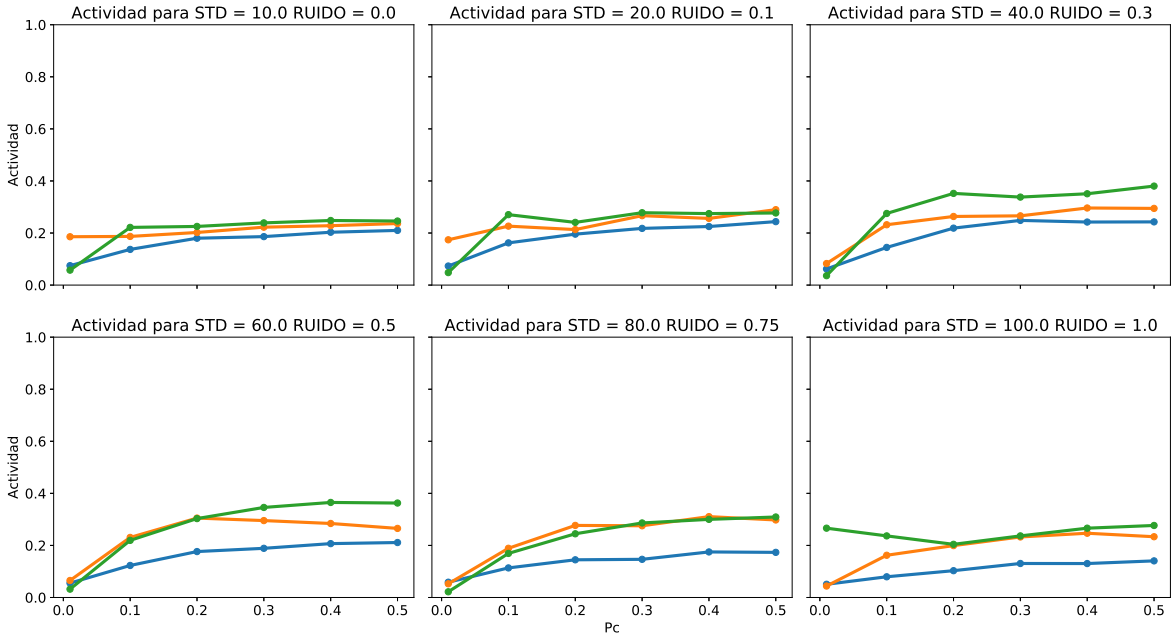

Figura 4.3: Nivel de activación medio de las neuronas KCs para los diferentes conjuntos de patrones gaussianos de dificultades variables en función del solapamiento entre las clases (consultar Sección 3.3.2. En cada gráfico se muestra el nivel de activación para los tres niveles de actividad utilizados y los diferentes valores de p_c en el intervalo $[0{,}01, 0{,}5]$.

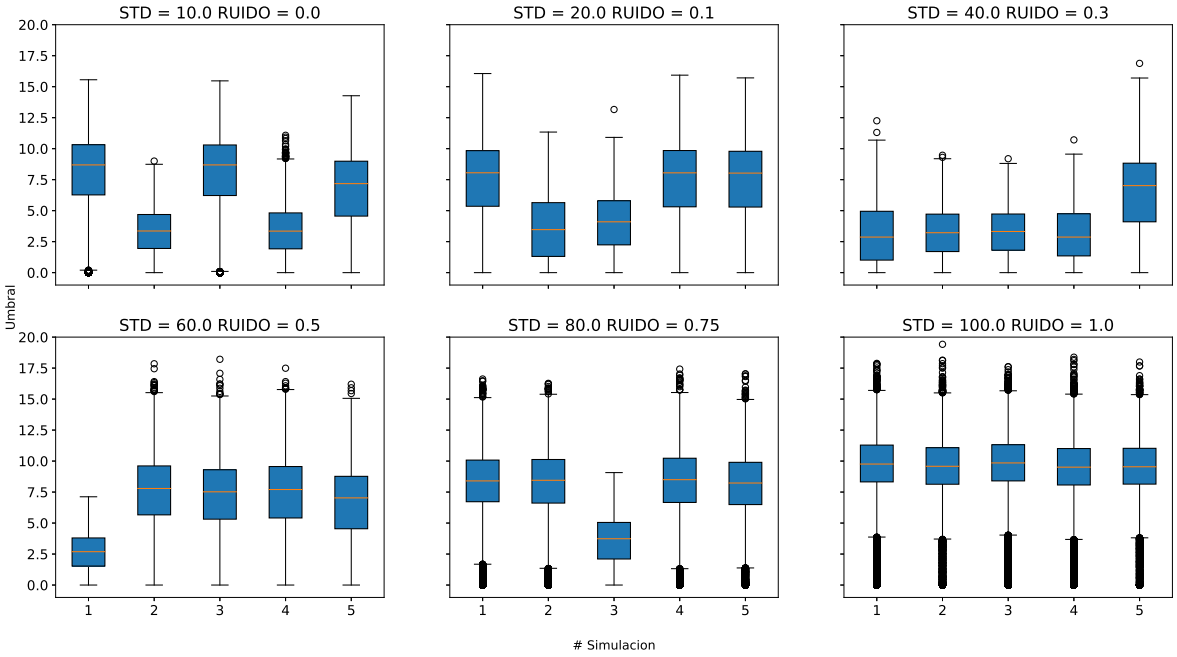

Figura 4.4: Distribución de umbrales neuronales óptima para los diferentes conjuntos de patrones gaussianos y los parámetros $p_c = 0{,}1$ y $s = 0{,}1$. Las distribuciones que se muestran se corresponden con cada una de las cinco iteraciones de validación cruzada de una simulación del modelo por cada conjunto de patrones.

lo que son necesarios valores de umbrales más altos en combinación con otros inusualmente bajos o altos para regular el gasto energético y conseguir el nivel de actividad requerido mediante el parámetro s.

Las preferencias respecto a las distribuciones de umbrales mencionadas en el párrafo anterior también se pueden analizar desde el punto de vista de su efecto en la capacidad de discriminación de la red. En los patrones de mayor complejidad, la presencia de una mayoría de neuronas con un umbral alto y, por tanto, un comportamiento especialista por el que sólo se activan cuando reciben un determinado número de estímulos concretos facilita la codificación dispersa de la información y mejora el reconocimiento de los patrones, aunque siguen siendo necesario un pequeño número de neuronas con umbrales más bajos y comportamiento generalista para extraer las características generales de los patrones. En los casos en los que la complejidad de los patrones es más baja, como ocurre en los gráficos de la fila superior de la Figura 4.4, no es tan necesario contar con gran cantidad de neuronas especialistas, por lo que existen soluciones en las que las neuronas generalistas predominan. Por tanto, el algoritmo desarrollado es capaz de encontrar la distribución de umbrales más ventajosa en función de la complejidad de forma que los resultados que se han obtenido son consistentes con el balance entre neuronas generalistas y especialistas necesario para el reconocimiento de patrones planteado en [5, 6, 7].

Para tener una visión más general del comportamiento de las distribuciones de umbrales no sólo en el caso óptimo, también se presenta la distribución del umbral medio, que se ha obtenido calculando la media de la distribución de umbrales de cada una de las cinco iteraciones de validación cruzada para diez simulaciones del modelo por cada nivel de complejidad. Los resultados se muestran en la Figura 4.5 para las combinaciones de $p_c = 0{,}1$ y $p_c = 0{,}4$ con los tres niveles de actividad y todos los niveles de complejidad.

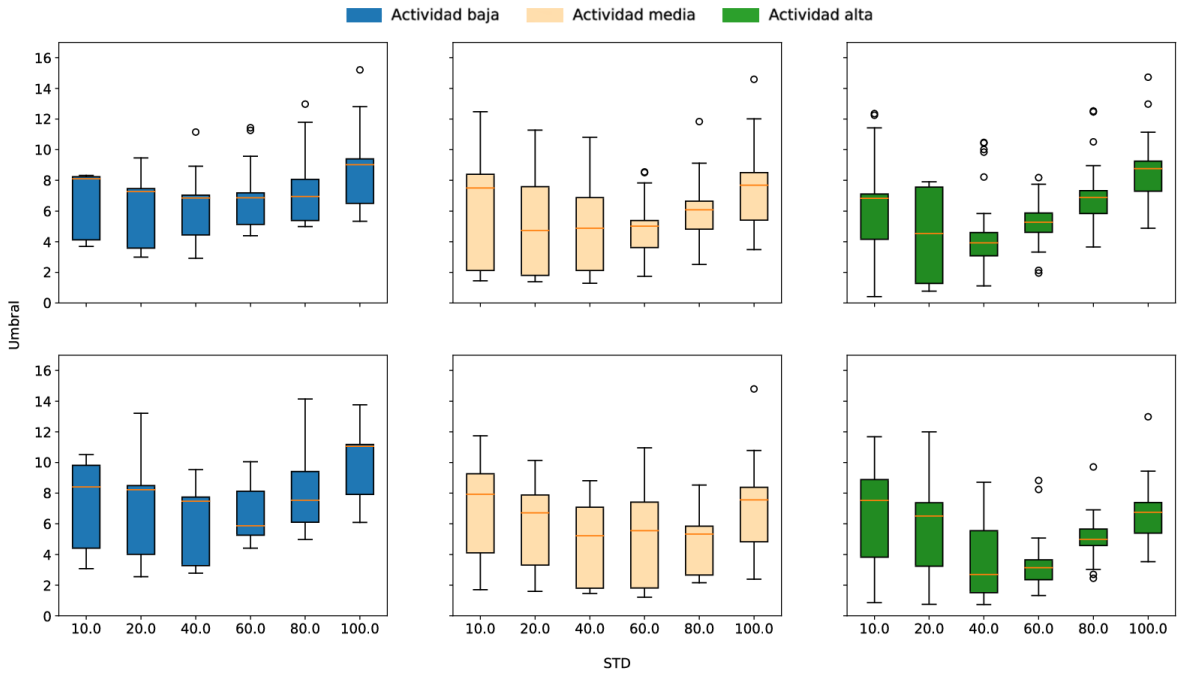

Figura 4.5: Distribución del umbral medio para las diferentes combinaciones de niveles de actividad, complejidad de los patrones con $p_c = 0{,}1$ en la línea superior y $p_c = 0{,}4$ en la fila inferior. Las distribuciones se han obtenido calculando la media de la distribución de umbrales de cada una de las cinco iteraciones de validación cruzada para diez simulaciones del modelo por cada nivel de complejidad.

La variabilidad de las distribuciones del umbral medio de las soluciones en función de la complejidad de los patrones no es muy fuerte y sigue el mismo comportamiento descrito anteriormente para la Figura 4.4: los patrones con complejidad intermedia presentan distribuciones del umbral medio centradas en valores bajos, los patrones de complejidad alta prefieren valores altos y los patrones de complejidad baja presentan distribuciones centradas en valores intermedios entre estos dos extremos debido a su falta de preferencia.

En cuanto a los diferentes niveles de actividad, puede verse cómo las distribuciones de umbrales se centran en valores mayores o menores dependiendo de el nivel que se quiera conseguir. Así, para el nivel de actividad bajo, las distribuciones se centran en torno a umbrales mayores y descienden para los niveles medio y alto. Lo mismo ocurre si se comparan las distribuciones para $p_c = 0{,}1$ y $p_c = 0{,}4$. Además, en este caso hay que tener en cuenta que valores más altos de p_c aumentan el flujo de información, lo que puede provocar que las desviaciones estándar de las distribuciones para $p_c = 0{,}4$ aumenten ligeramente con respecto a $p_c = 0{,}1$, como ocurre en el caso de los patrones de complejidad intermedia y alta.

4.4. Resultados para patrones MNIST

En esta sección se exponen los resultados obtenidos clasificando los patrones MNIST con el modelo desarrollado. Como en el caso de los patrones gaussianos, se muestran el error de clasificación y el nivel de actividad asociado a cada combinación de parámetros y las distribuciones de umbrales ajustadas por el algoritmo de aprendizaje. Además, en esta sección se introduce una comparación entre el error de clasificación obtenido utilizando y sin utilizar el mecanismo

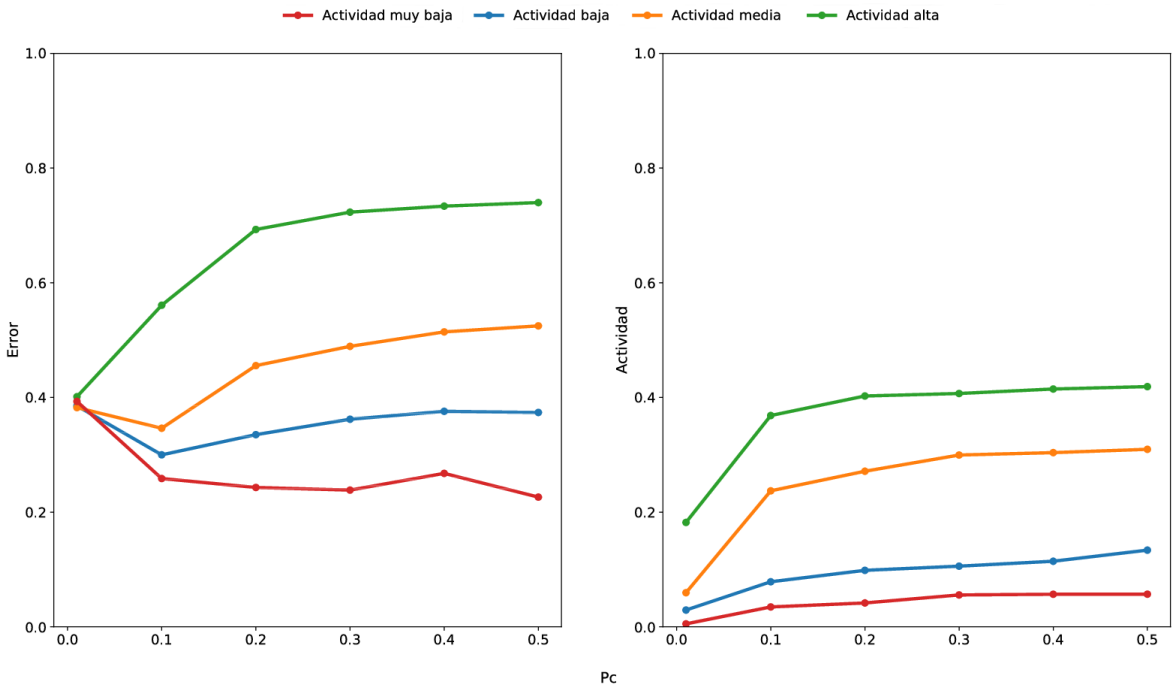

Figura 4.6: Error de clasificación y activación media de las neuronas KCs promediado para los patrones MNIST. Los resultados que se muestran son el promedio de 10 simulaciones.

de control de ganancia del modelo para comprobar cuál es su influencia en su capacidad de discriminación.

4.4.1. Error de clasificación y nivel de actividad

En la figura Figura 4.6 se muestran el error de clasificación obtenido para los patrones del MNIST y la actividad media de las neuronas KCs para los cuatro niveles de actividad y valores de p_c simulados (los valores de los parámetros pueden consultarse en el Cuadro 4.1). Debido a la complejidad de los patrones MNIST y a que el error de clasificación para el nivel de actividad bajo con $s = 0{,}1$ son superiores al 30 %, se decidió incluir un nivel de actividad más bajo con $s = 0{,}05$ y una activación media menor o igual al 5 % para comprobar si este nivel supone alguna ventaja en el caso de estos patrones.

Los resultados obtenidos siguen la línea de los mostrados para los patrones gaussianos en la Sección 4.3. De nuevo, los niveles de actividad bajo y muy bajo obtienen mejores resultados, consiguiendo un error de clasificación medio del 23,06 % y del 29.50 % respectivamente, lo que contrasta con los errores obtenidos por los niveles medio y alto, un 37,05 % en el primer caso y un 64,19 % en el segundo. El error mínimo se alcanza para el nivel de actividad muy bajo, sin que haya importantes diferencias entre los valores de $p_c \geq 0{,}1$. Por otro lado, al igual que en el caso de las gaussianas, se pueden descartar los valores de p_c muy bajos como $p_c = 0{,}01$, salvo en el caso del nivel de actividad alto, donde sí es más ventajoso.

Estos resultados también están en consonancia con el balance entre la cantidad de información que pasa de las neuronas de entrada PNs a la capa de las KCs y la calidad del código disperso utilizado para representarla. El nivel de actividad bajo da mejores resultados en combinación con valores pequeños de p_c que dejan pasar una cantidad más reducida de información sobre los patrones, eliminando ruido y haciendo más fácil su reconocimiento. En el caso del nivel

de actividad muy bajo, la robustez del código disperso utilizado hace que el error se mantenga sin variaciones importantes con independencia del valor de p_c.

Sin embargo, para los niveles de actividad medio y alto, el comportamiento es diferente al observado en los patrones gaussianos. Mientras que para estos el error se reducía a medida que p_c aumentaba, en el caso de MNIST, el error aumenta a medida que lo hace p_c y el error mínimo se da para $p_c \leq 0{,}1$. La explicación a este comportamiento podría estar en la propia estructura de los patrones. Los patrones del MNIST se caracterizan por tener cierta cantidad de neuronas de entrada que nunca se activan porque pertenecen a los marcos alrededor de los números y diferentes zonas que sólo se activan para ciertos dígitos, lo que reduciría la cantidad de información necesaria para poder distinguir unas clases de otras y, por tanto favorecería valores de p_c pequeños. Los valores mayores sólo contribuirían a dejar pasar información innecesaria o redundante que entorpecería el reconocimiento.

Por tanto, para los patrones MNIST, las configuraciones del modelo que dan mejores resultados son las que presentan una actividad muy baja, no superior al 5 % en este caso, y los valores de $p_c \in [0{,}1, 0{,}5]$, sin que haya grandes variaciones entre ellos. De nuevo, el modelo converge a un estado que es coherente con lo propuesto en otros trabajos teóricos y estudios biológicos [9, 49, 50, 41, 55] y lo hace gracias al control de la actividad en las KCs por medio de la distribución de sus umbrales.

4.4.2. Distribución de umbrales neuronales

En la Figura 4.7 se muestran las distribuciones de umbrales óptimos para los niveles de actividad bajo y muy bajo y $p_c = 0{,}1$. Se han elegido estos valores porque se corresponden con los valores más bajos de error de clasificación.

Como puede verse, las distribuciones de umbrales presentan una forma parecida a las que obtuvieron para los patrones gaussianos de mayor complejidad (Figura 4.4 inferior derecha). En este caso, los umbrales siempre se encuentran dentro del intervalo $[0{,}0 - 15]$ para el caso de la actividad baja, lo que coincide con los patrones gaussianos, y $[0{,}0 - 25{,}0]$ para el caso de la actividad muy baja.

Por lo general, las distribuciones no presentan grandes variaciones y se caracterizan por estar concentradas en torno a unos pocos valores de umbrales. Estos valores del umbral son altos (~ 10 para actividad baja y ~ 12 para actividad muy baja), lo que favorece un comportamiento especialista de las neuronas KCs y facilita la codificación dispersa, como ya se indicó anteriormente. Los valores de umbral neuronal excepcionalmente bajos (por debajo de 5), que se corresponderían con el comportamiento muy generalista, sólo aparecen como valores atípicos. Por tanto, para la clasificación de los patrones MNIST, el modelo muestra preferencia por una población de neuronas mayormente especialistas combinadas con muy poca cantidad de neuronas generalistas o muy especializadas, que ayudarían al reconocimiento de los patrones y a regular la actividad. En este caso el modelo también converge por sí mismo a distribuciones de umbrales heterogéneas que son coherentes con lo propuesto en [7], donde se muestra que en el caso de patrones de complejidad elevada como el MNIST, son más ventajosas poblaciones de neuronas casi exclusivamente especialistas.

En la Figura 4.8 también se muestran las distribuciones del umbral medio para todas las combinaciones de niveles de actividad y p_c. Como en el caso de los patrones gaussianos, las distribuciones se han obtenido calculando la media de las distribuciones de umbrales obtenidas para cada una de las cinco iteraciones de validación cruzada en diez simulaciones.

En este caso, se observa una fuerte variación en los valores en torno a los que se centran estas distribuciones en función de los niveles de actividad. Para los niveles alto y medio predominan

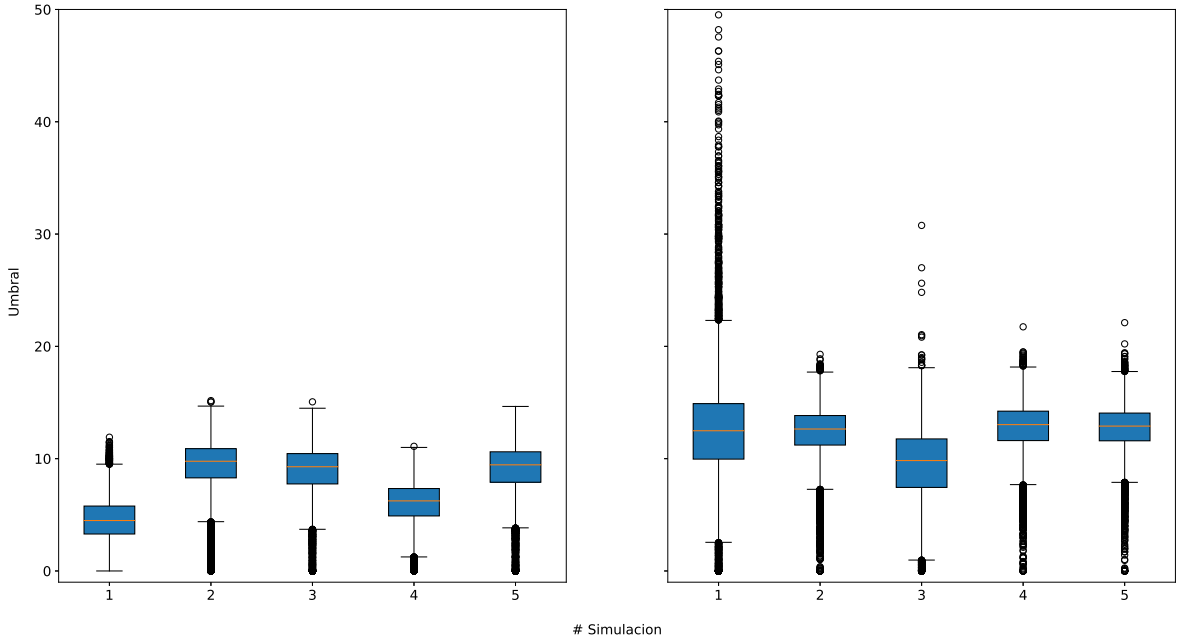

Figura 4.7: Diagramas de cajas para la distribución de umbrales óptimas para actividad baja con $p_c = 0{,}1$ a la izquierda y actividad muy baja con $p_c = 0{,}1$ a la derecha. Se han elegido estos valores de p_c al ser los que consiguen un error de clasificación más bajo. Se muestran las distribuciones ajustadas para cada una de las cinco iteraciones de validación cruzada en una simulación del modelo.

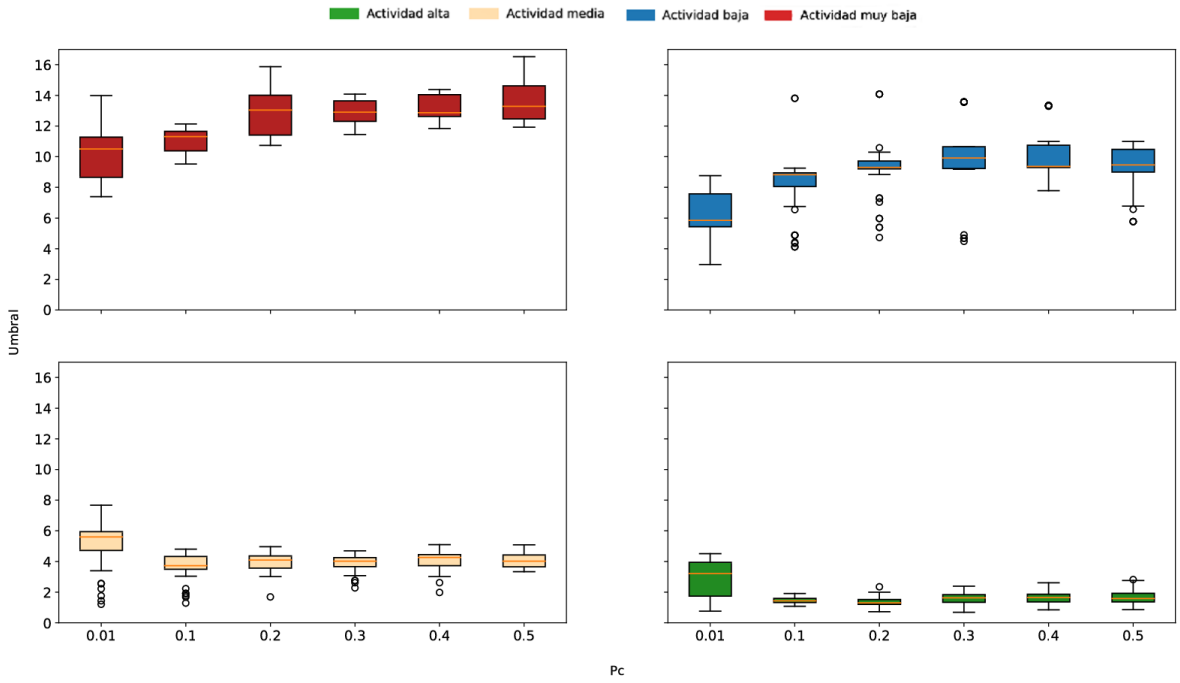

Figura 4.8: Diagrama de la distribución del umbral neuronal medio para los patrones MNIST con los diferentes niveles de actividad y valores de p_c. Los resultados se han obtenido mediante 10 simulaciones por cada combinación con validación cruzada con 5 iteraciones.

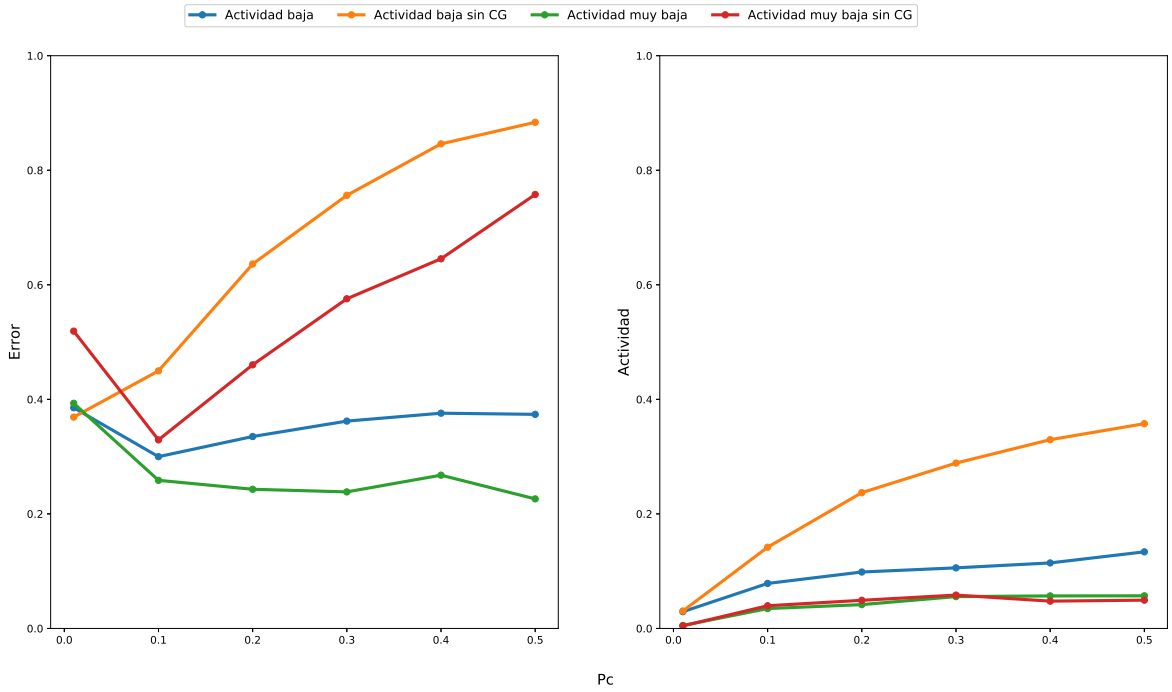

Figura 4.9: Comparación entre el error de clasificación y el nivel de actividad obtenido utilizando o no el mecanismo de control de ganancia. Las diez simulaciones por cada punto se han realizado para el nivel bajo y muy bajo de actividad, al ser los que obtienen mejores resultados, y todos los valores de p_c considerados.

unos valores medios de umbral muy bajos, lo que llevaría a un comportamiento generalista de casi todas las neuronas de la capa, por lo que el código disperso no puede generarse en estos casos, lo que explica los altos errores de clasificación de los patrones con estas configuraciones. Por el contrario, en el caso de la actividad muy baja, las soluciones presentan umbrales medios unas cinco veces más elevados que en el caso anterior, lo que favorece la eficiencia energética, el comportamiento especialista y el código disperso.

4.4.3. Influencia del control de ganancia

Para comprobar el efecto que el método de control de ganancia sencillo que se ha introducido en el modelo tiene sobre el error de clasificación y la actividad media de las neuronas KCs, realizamos diez simulaciones para los niveles de actividad bajo y muy bajo combinados con todos los valores de p_c. La comparativa puede observarse en la Figura 4.9.

Como se muestra en los gráficos, el control de ganancia tiene una influencia importante en el control del gasto energético y la reducción del error de clasificación. Cuando no hay control de ganancia, el error se incrementa de un 30 % a un 40 % en comparación con los resultados obtenidos cuando sí se incluye el control de ganancia. Además, el error muestra una fuerte tendencia al alza a medida que p_c aumenta. Por otro lado, en el gráfico de actividad puede verse que para el nivel bajo, la actividad se descontrola y crece hasta el 40 %, lo que desemboca en un error muy alto. En el caso de la actividad muy baja, sin control de ganancia se mantiene en los niveles esperados, aunque el error de clasificación sigue siendo mucho más alto, lo que en este caso se debería a la incapacidad del modelo para encontrar los pesos y umbrales adecuados.

En conclusión, estos resultados resaltan la importancia de incluir un método de control de

ganancia en el modelo que permita que los patrones de diferentes clases puedan ser representados de forma estándar con independencia de su magnitud o de la cantidad de neuronas de entrada que activen.

4.5. Resultados para los patrones de odorantes captados por narices electrónicas

Uno de los mayores problemas que suceden en los sistemas de detección de odorantes es que los sensores de las narices electrónicas que se utilizan para captarlos se deterioran y contaminan con el tiempo, lo que resulta en que las mediciones que realizan para los mismos odorantes en idénticas condiciones experimentan variaciones graduales que se conocen como deriva. Con el objetivo de poder desarrollar técnicas de clasificación capaces de solucionar el problema de la deriva, en [68] se publicó una base de datos de odorantes capturados con los mismos sensores durante tres años, periodo en que se contaminaron y deterioraron. Estos datos constan de 128 atributos y 6 clases que se corresponden con seis odorantes a diferentes concentraciones, como se describió en la Sección 3.3.3. Los datos están divididos en 10 conjuntos diferentes $(B_1, B_2, ..., B_{10})$, agrupados por proximidad temporal, como se observa en el Cuadro 4 de [68]. De esta forma, los patrones correspondientes al mismo conjunto, presentan una complejidad baja y son fáciles de clasificar, pero si se intenta clasificar datos de un conjunto entrenando un clasificador con otro conjunto diferente, debido a la deriva de los datos, se convierte en una tarea muy compleja, como se observa en la Figura 4 de [68].

Teniendo en cuenta lo anterior, el objetivo de esta sección es clasificar con el modelo del sistema olfativo desarrollado estos conjuntos de datos siguiendo algunas de las estrategias de [68] para minimizar el efecto de la deriva en la clasificación de los odorantes con SVMs y comparar los resultados. La hipótesis que se plantea es que, si el sistema olfativo de los insectos hace operaciones similares a las que se llevan a cabo en una SVM para clasificar como se plantea en [41, 70], los resultados obtenidos con el modelo serán iguales o mejores. Esto se analizará en más detalle en Sección 4.5.4.

Para poder llevar a cabo la comparación entre los resultados obtenidos clasificando los odorantes por las SVMs y el modelo del sistema olfativo, en primer lugar es necesario encontrar la combinación de parámetros de este último que logra los mejores resultados, así como caracterizar el tipo de soluciones a las que llega el sistema mediante la inspección de las distribuciones de umbrales neuronales. Estos análisis se presentan en los siguientes apartados. Debido a que las variaciones temporales que experimentan los patrones de los diferentes conjuntos de datos, se ha optado en este caso por emplear validación simple en todas las simulaciones para evitar que el modelo tenga información directa sobre los patrones que se van a clasificar.

4.5.1. Error de clasificación y nivel de actividad

En primer lugar, para comprobar cuál es la tasa de acierto del modelo para los patrones de odorantes y encontrar la combinación de parámetros que mejor resultados obtiene, se entrena el modelo utilizando el conjunto de patrones B_1 y se clasifican los patrones pertenecientes al conjunto B_2, para los niveles de actividad y p_c utilizados en los apartados anteriores. Basándonos en los resultados de la clasificación de estos patrones para SVMs en la Figura 5 [68], el error no debería ser superior al 25 %. En la Figura 4.10 se presentan los resultados obtenidos.

En este caso los resultados también son similares a los obtenidos en los apartados anteriores. El error de clasificación mínimo vuelve a alcanzarse para el nivel de actividad más bajo, sin variaciones fuertes dependientes de p_c, aunque los valores más elevados conllevan una mejora

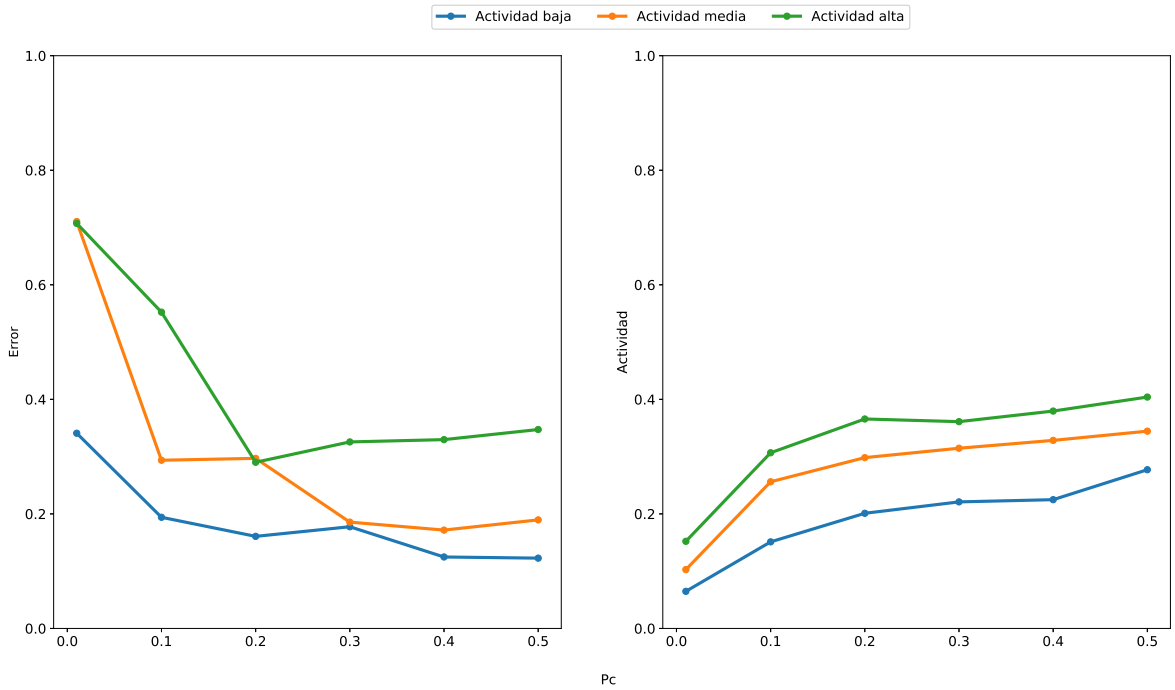

Figura 4.10: Error de clasificación y activación media de las neuronas KCs para las diferentes combinaciones de nivel de actividad con p_c. El modelo se entrena con el conjunto de B_1 y se clasifican los patrones en el conjunto B_2.

de aproximadamente un 3 %-5 %, hay que tener en cuenta que el nivel de activación media para estos casos supera o es muy cercano al 20 %. Para los niveles de actividad medio y alto, el balance entre la información que se proyecta a la capa de las KCs y el grado de dispersión del código utilizado para representarla se da para valores medios de p_c, entre 0.2 y 0.3, mostrando una tendencia creciente a medida que p_c aumenta desde estos valores. Por último, de nuevo los valores de $p_c < 0{,}1$ son los que peores resultados obtienen, ya que apenas permiten el paso de la información.

Teniendo en cuenta todo lo anterior, para realizar el resto de simulaciones y comparar los resultados con los obtenidos en [68] para la clasificación de los 10 conjuntos de odorantes, se utiliza la combinación de parámetros $s = 0{,}1$ para conseguir un nivel de actividad bajo y $p_c = 0{,}1$, ya que con esta combinación se obtiene un error de clasificación bajo y el nivel de actividad se mantiene en torno al 10 %.

4.5.2. Distribución de umbrales neuronales

En la Figura 4.11 se muestran las distribuciones de umbrales óptimas para cinco simulaciones diferentes para los valores de $s = 0{,}1$ y $p_c = 0{,}1$ seleccionados a partir de los resultados del apartado anterior. A diferencia de los conjuntos de patrones anteriores como el MNIST o las gaussianas con una mayor complejidad, el modelo muestra preferencia por las distribuciones centradas en torno a valores de umbral bajos en combinación con ciertos valores de umbral más altos. Por tanto, para estos patrones, la mayor parte de las neuronas KCs tienen un comportamiento generalista y una pequeña parte de la población presenta comportamiento especialista. Este tipo de distribución es propia de patrones que presentan una complejidad intermedia, como se vio en los resultados de las gaussianas (Sección 4.3.2). Para estos patrones, es más ventajoso mantener un balance entre neuronas especialistas y generalistas [7].

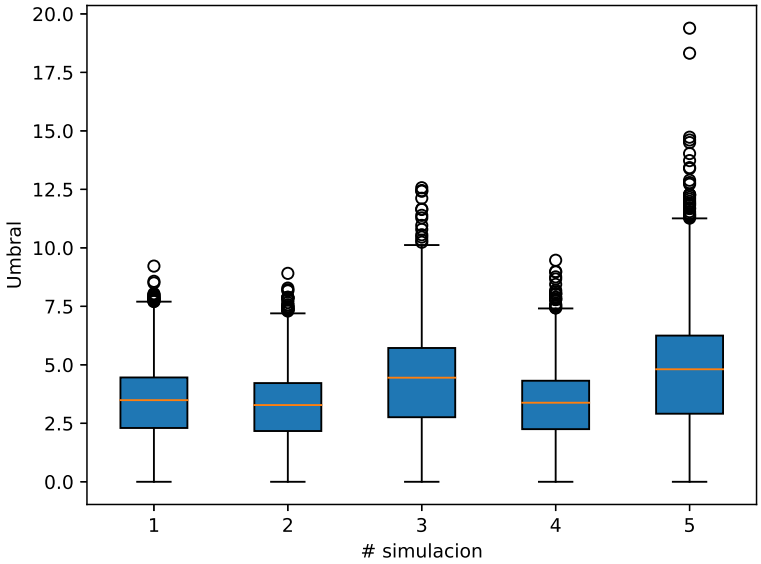

Figura 4.11: Distribuciones de umbrales óptimas para cinco simulaciones con el nivel de actividad bajo y $p_c = 0{,}1$. El modelo se entrena utilizando el conjunto de odorantes 1 y se evalúa clasificando el 2.

En la Figura 4.12 se muestra la distribución del umbral neuronal medio en diez simulaciones diferentes para todas las combinaciones de nivel de actividad y valor de p_c. Como puede verse, la variación de las soluciones es pequeña y en todos los casos el modelo siempre converge a distribuciones cuyo umbral medio es similar. También confirma que todas las combinaciones de parámetros dan como resultado distribuciones cuyo umbral medio es bajo, por lo que incluirán una mayoría de neuronas generalistas. Por último, mientras que para el nivel de actividad alto no hay demasiada variación en los valores de los umbrales en función de p_c, para la actividad media y baja se observa cómo las distribuciones se centran en torno a valores más bajos a medida que aumenta p_c, lo que indica que en estos casos las neuronas muy generalistas son más ventajosas.

4.5.3. Influencia del control de ganancia

Para comprobar y comparar los resultados obtenidos para los patrones MNIST con y sin el método de control de ganancia del modelo (Sección 4.4.3), se han llevado a cabo simulaciones del modelo para los patrones de odorantes con el nivel de actividad bajo y los diferentes valores de p_c sin control de ganancia. La comparativa se muestra en la Figura 4.13.

Como puede verse, los resultados confirman los obtenidos en el caso del MNIST. La utilización del mecanismo de control de ganancia asegura la representación estándar de los patrones con independencia de la cantidad de neuronas o de actividad que provoquen en la capa de entrada y permite una mejora en el error de clasificación de hasta un 40 % para los valores de p_c más altos y contribuye al control de la actividad en la capa de las KCs, reduciéndolo hasta en un 20 %.

4.5.4. Comparación con SVMs

A continuación se muestran los resultados obtenidos clasificando el resto de conjuntos de datos con los parámetros $p_c = 0{,}1$ y $s = 0{,}1$ utilizando dos de las configuraciones propuestas en [68] para minimizar los efectos de la deriva en la clasificación. De esta forma, los resultados

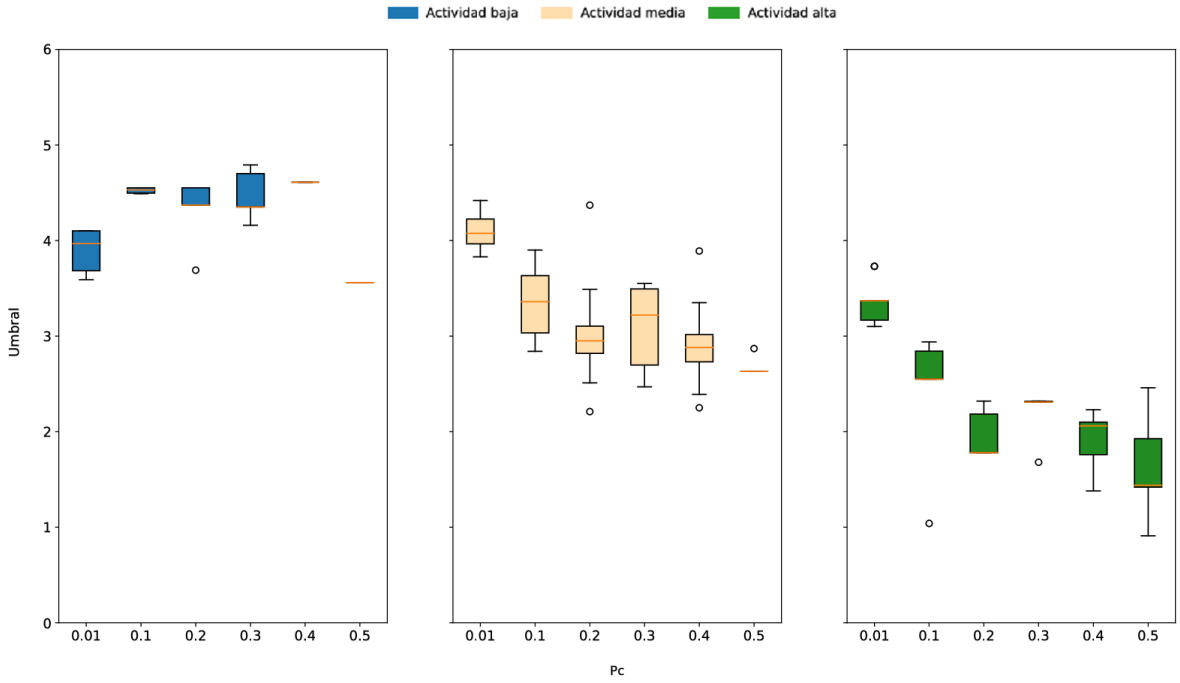

Figura 4.12: Distribuciones del umbral medio en diez simulaciones diferentes para todas las combinaciones de nivel de actividad y p_c.

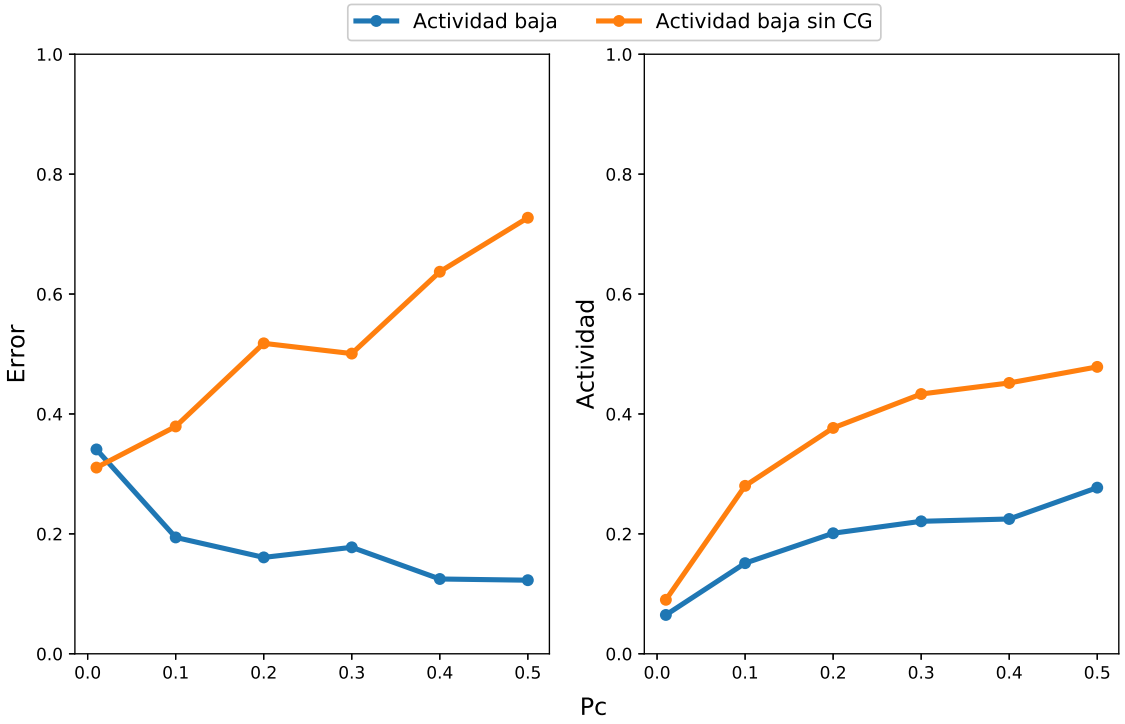

Figura 4.13: Resultados obtenidos para el nivel bajo de actividad con los diferentes valores de p_c con y sin incluir el control de ganancia. El modelo se ha entrenado con los patrones del conjunto de odorantes B_1 y se ha evaluado con el conjunto B_2.

obtenidos mediante el modelo podrán compararse con los obtenidos clasificando con una SVM, extraídos a partir de los resultados presentados en la Figura 5 de [68] . Los resultados para ambas configuraciones se muestran en la Figura 4.14.

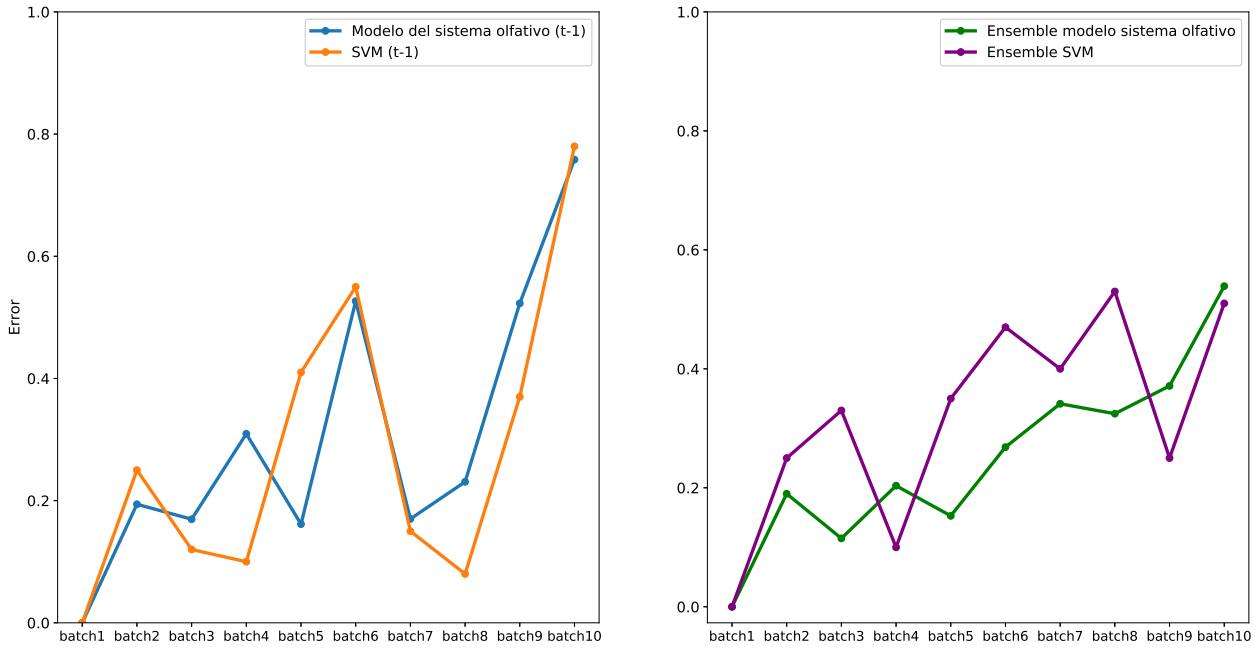

Figura 4.14: Comparación entre los resultados obtenidos clasificando con una SVM o con el modelo de sistema olfativo. La figura de la derecha corresponde a la primera configuración y la figura de la izquierda muestra los resultados para la segunda. Los resultados de clasificación para las SVMs se han obtenido a partir de la Figura 5 de [68].

En la primera de las configuraciones, para clasificar cada uno de los conjuntos de datos B_t, se entrena un clasificador empleando el conjunto $B_{(t-1)}$. Debido a que la deriva tiene un mayor impacto en las mediciones conforme estas se alejan en el tiempo, al tratarse en este caso de conjuntos de datos consecutivos, se supone que el efecto de la deriva es pequeño y no hay demasiados cambios entre los patrones de ambos conjuntos, por lo que se espera que el error de clasificación sea relativamente bajo. Por tanto, esta configuración supone un buen punto de referencia y cualquier otro método que mejore los errores conseguidos con esta configuración constituye un resultado positivo [68]. Como se puede ver en Figura 4.14 izquierda, la previsión de errores bajos se cumple para la mayoría de conjuntos salvo para B_6, B_9 y B_{10}. B_{10} es el más problemático debido a que contiene mediciones que se tomaron más de seis meses después de elaborar el B_9, por lo que el efecto de la deriva es muchísimo mayor. En general, las SVMs obtienen mejores resultados que el modelo del sistema olfativo, aunque en algunos conjuntos como el B_1, B_2, B_5, B_6 y B_{10} el rendimiento de este último es igual o mejor que el conseguido mediante la SVM.

En la segunda configuración, para clasificar un conjunto de datos B_t se entrenan $t-1$ clasificadores con los $t-1$ conjuntos anteriores a B_t. Por ejemplo, si el conjunto a clasificar es B_4, se entrenan dos clasificadores con los conjuntos B_3, B_2 y B_1 respectivamente. Una vez se han entrenado los $t-1$ clasificadores, se construye un *ensemble* agrupando todos ellos. A la hora de clasificar los patrones de B_t, se asigna la clase maś votada por los clasificadores del *ensemble*, teniendo todos los votos el mismo peso. Esta configuración presenta la ventaja de que en la

clasificación de los patrones interviene información procedente de todos los conjuntos anteriores, tanto los más lejanos temporalmente, como los más cercanos, por lo que se espera que el sistema generalice mejor y se compense el efecto de la deriva de los datos.

En consonancia con lo anterior, en la Figura 4.14 derecha se puede observar que la segunda configuración que incluye el *ensemble* mejora los resultados de la primera configuración para todos los casos, tanto utilizando SVMs como utilizando el modelo del sistema olfativo, lo que constituye un buen resultado. Centrándonos solo en la segunda configuración, el modelo consigue mejorar o igualar los resultados obtenidos utilizando una SVM con la misma configuración para todos los conjunto salvo el B_4 y el B_9.

Por tanto, a partir de los resultados para ambas configuraciones, se muestra que la capacidad de discriminación del modelo desarrollado sería parecido al de una SVM ya que el modelo realizaría operaciones equivalentes a las que se llevan a cabo en este método, como se plantea en [41, 70]. Además de esto, es capaz de evitar en gran medida los efectos de la deriva que sufren los patrones de odorantes y conseguir buenos resultados de clasificación para la mayoría de conjuntos cuando se ha utilizado una de las estrategias propuestas en [68] basada en un *ensemble* de clasificadores.

5

Conclusiones y trabajo futuro

5.1. Conclusiones

El proyecto desarrollado ha cumplido con el objetivo principal que se planteó de desarrollar un modelo del sistema olfativo del insecto (en concreto, de la langosta voladora), que permitiese explorar el impacto en el funcionamiento del sistema de varias de las estrategias computacionales que emplea, como el código disperso y una distribución de umbrales heterogénea en las KCs. Para ello, se han seguido los siguientes pasos:

- Se ha realizado un estudio sobre el sistema olfativo de los insectos, los diferentes modelos que se han planteado para reproducir su comportamiento y estudiarlo y los valores que se han estimado para varios de sus parámetros más importantes. Dicho estudio se ha presentado en el estado del arte de este proyecto.
- Se ha desarrollado un modelo del sistema olfativo de los insectos basado en redes neuronales que incluye el diseño de un algoritmo de aprendizaje capaz de ajustar los umbrales de disparo de las neuronas KCs y MBONs, así como controlar el nivel de actividad en las neuronas KCs y la probabilidad de conexión ente las PNs y las KCs. Todos estos parámetros influyen en la forma en que se codifica la información en el modelo y permiten investigar la generación del código disperso y su influencia a la hora de mejorar la capacidad de discriminación de patrones de la red.
- El modelo desarrollado se ha validado mediante tres problemas de clasificación que incluyen patrones con diferentes niveles de complejidad, obteniendo resultados satisfactorios para todos ellos.
- Los resultados obtenidos con el modelo se han comparado con los obtenidos en otras investigaciones llevadas a cabo sobre el sistema olfativo de los insectos.

La conclusión principal que se ha obtenido a partir de los resultados de las simulaciones del modelo para los diferentes problemas de clasificación es que el algoritmo de aprendizaje desarrollado es capaz de regular la actividad de las neuronas KCs a partir del ajuste de sus umbrales de disparo y obtiene mejores resultados cuando converge a un estado similar al encontrado en la biología (actividad baja en las KCs de $\sim 10\,\%$ y valor moderado de p_c) [55, 9]. Además, también

es capaz de ajustar las características de la distribución de umbrales de disparo en las KCs para adaptarla a la complejidad de los patrones que se están clasificando [7]. Por tanto, el umbral neuronal tiene una gran importancia a la hora de determinar la capacidad de discriminación del sistema y controlar la generación del código disperso para la representación de la información. Otras de las conclusiones más importantes que se han obtenido tras el análisis de los resultados y que se derivan de la anterior, son:

- El balance entre la cantidad de información que pasa de las PNs a las KCs, controlada por la probabilidad de conexión entre PNs y KCs, y la calidad del código disperso utilizado para la representación de dicha información, determinada por el nivel de actividad en las KCs, es otro de los factores que más influencian la capacidad de discriminación del sistema.
- El balance entre el nivel de actividad de las neuronas KCs y la probabilidad de conexión entre PNs y KCs que resulta más ventajoso varía con la complejidad de los patrones clasificados, pero se ha mostrado que el nivel de actividad baja siempre conduce a mejores resultados, ya que favorece la generación de un código disperso robusto que facilita la tarea de clasificación [32, 31].
- En cuanto al valor de la probabilidad de conexión, los resultados nos permiten descartar valores de p_c inferiores a 0.1, al menos cuando se utiliza una matriz estocástica para modelar la conectividad entre las dos capas. Aunque no existe mucha variación en el error de clasificación para los diferentes valores de p_c analizados, el valor $p_c = 0{,}1$ da buenos resultados y permite que la actividad en las KCs sea $10\,\%$, lo que es consistente con los hechos biológicos [55]. Los resultados tampoco permiten descartar el valor de $p_c = 0{,}5$ que se ha aceptado en los últimos tiempos [9, 49, 50].
- En cuanto a la distribución de umbrales neuronales, el modelo siempre converge a distribuciones heterogéneas en las que se combinan diferentes cantidades de neuronas con umbrales bajos de comportamiento generalista y neuronas de umbrales altos con comportamiento especialista que serían la base del código disperso. Los resultados sugieren que el tamaño de cada una de estas poblaciones estaría relacionado con la complejidad de los patrones a clasificar, lo que es coherente con las conclusiones obtenidas en los estudios previos llevados a cabo en el GNB [5, 6, 7].
- También se ha mostrado que el mecanismo de control de ganancia ayuda a representar los patrones de una forma más estándar y, por tanto, tienen un gran impacto en el sistema a la hora de mejorar el error de clasificación y contribuir al control de la actividad en la capa de las KCs [43, 46, 47].
- Se ha comprobado el rendimiento del modelo clasificando datos de odorantes capturados por narices electrónicas [68], lo que constituye una tarea más parecida a la que el sistema tendría que resolver en la naturaleza. Los resultados obtenidos son comparables a los conseguidos mediante el uso de SVMs [68], lo que es coherente con las comparaciones establecidas entre las operaciones llevadas a cabo por el sistema olfativo en sus redes neuronales y por una SVM [41, 70].

Por último, algunos de los resultados de este proyecto han sido aceptados para su publicación en los congresos CNS2018 [11] e ICANN2018 [12].

5.2. Trabajo futuro

El presente proyecto se seguirá desarrollando en el futuro, con el objetivo de iniciar una tesis doctoral, mediante las tareas que se exponen a continuación:

- Publicación del modelo desarrollado y los resultados obtenidos en una revista científica.
- Incorporación al modelo de un mecanismo de control de ganancia más elaborado e inspirado en la biología.
- Obtención de la distribución de umbrales en las KCs a partir de datos procedentes del sistema biológico para comparar los resultados con los obtenidos por el modelo.
- Adaptación del algoritmo de aprendizaje de umbrales neuronales al aprendizaje no supervisado para que tenga un comportamiento más realista.

Glosario

- **GNB**: Grupo de Neurocomputación Biológica
- **UAM**: Universidad Autónoma de Madrid
- **LA**: lóbulo antenal de la langosta.
- **CF**: cuerpo fungiforme del sistema olfativo de la langosta.
- **PN**: *Proyection Neuron*. Neuronas ubicadas en el lóbulo antenal de la langosta.
- **KC**: *Kenyon Cell*. Neuronas localizadas en el cuerpo fungiforme del sistema olfativo de la langosta.
- **GGN**: gigante GABAérgico. Neurona que forma parte del sistema olfativo de la langosta y actúa sobre las KCs para regular su excitabilidad mediante inhibición periódica.
- **MBON**: *Mushroom Body Output Neuron*. Población de neuronas que reciben la información de las KCs y son las últimas responsables de la discriminación de los odorantes.
- **MSE**: *Mean Squared Error*. Función de coste que calcula el error cometido por un estimador como el promedio de los errores entre los valores estimados y los valores reales al cuadrado.

Bibliografía

[1] J. A. Sigüenza. *Cómo funciona el cerebro*. Eudema, Madrid, 1993.

[2] A. Longstaff. *Neuroscience*. Springer, New York, 2000.

[3] Gabriel Kreiman. Neural coding: computational and biophysical perspectives. *Physics of Life Reviews*, 1(2):71–102, jul 2004.

[4] Björn Naundorf, Fred Wolf, and Maxim Volgushev. Unique features of action potential initiation in cortical neurons. *Nature*, 440(7087):1060–1063, apr 2006.

[5] Aarón Montero, Ramón Huerta, and Francisco Borja Rodríguez. Neuron Threshold Variability in an Olfactory Model Improves Odorant Discrimination. Lecture Notes in Computer Science, pages 16–25. Springer, Berlin, Heidelberg, jun 2013.

[6] Aarón Montero, Ramón Huerta, and Francisco Borja Rodríguez. Regulation of specialists and generalists by neural variability improves pattern recognition performance. *Neurocomputing*, 151(Part 1):69–77, mar 2015.

[7] Aaron Montero, Ramon Huerta, and Francisco B. Rodriguez. Stimulus space complexity determines the ratio of specialist and generalist neurons during pattern recognition. *Journal of the Franklin Institute*, 355(5):2951–2977, feb 2018.

[8] Marta García-Sanchez and Ramón Huerta. Design parameters of the fan-out phase of sensory systems. *Journal of Computational Neuroscience*, 15(1):5–17, jul 2003.

[9] Ron A Jortner, Sarah S Farivar, and Gilles Laurent. A Simple Connectivity Scheme for Sparse Coding in an Olfactory System. *Journal of Neuroscience*, 27(7):1659–1669, feb 2007.

[10] J. Pérez-Orive, O. Mazor, G. C. Turner, S. Cassenaer, R. I. Wilson, and G. Laurent. Oscillations and sparsening of odor representations in the mushroom body. *Science*, 297(5580):359–65, 2002.

[11] Jessica Lopez-Hazas, Aaron Montero, and Francisco Borja Rodriguez. Regulation of Neural Threshold in Kenyon Cells through their Sparse Condition improves Pattern Recognition Performance. *To appear in BMC Neuroscience*, 2018. To appear.

[12] Jessica Lopez-Hazas, Aaron Montero, and Francisco Borja Rodriguez. Strategies to enhace Pattern Recognition in Neural Networks based on the Insect Olfactory System. Springer, Berlin, Heidelberg, 2018. To appear.

[13] E D Adrian and Y Zotterman. The impulses produced by sensory nerve-endings: Part II. The response of a Single End-Organ. *The Journal of physiology*, 61(2):151–71, apr 1926.

[14] Susumu Hagiwara. Analysis of interval fluctuation of the sensory nerve impulse. *The Japanese Journal of Physiology*, 4:234–240, 1954.

[15] Nicholas J Priebe, Ferenc Mechler, Matteo Carandini, and David Ferster. The contribution of spike threshold to the dichotomy of cortical simple and complex cells. *Nature Neuroscience*, 7(10):1113–1122, oct 2004.

[16] T. Tateno and H.P.C. Robinson. Rate coding and spike-time variability in cortical neurons with two types of threshold dynamics. *Journal of Neurophysiology*, 95(4):2650–2663, apr 2006.

[17] Alan B. Saul and Allen L. Humphrey. Temporal-frequency tuning of direction selectivity in cat visual cortex. *Visual Neuroscience*, 8(4):365–372, apr 1992.

[18] Monty A. Escabí, Reza Nassiri, Lee M. Miller, Christoph E. Schreiner, and Heather L. Read. The Contribution of Spike Threshold to Acoustic Feature Selectivity, Spike Information Content, and Information Throughput. *Journal of Neuroscience*, 25(41):9524–9534, oct 2005.

[19] Nicholas J. Priebe and David Ferster. Inhibition, spike threshold, and stimulus selectivity in primary visual cortex. *Neuron*, 57(4):482–497, feb 2008.

[20] Wilent W. Bryan and Diego Contreras. Stimulus-Dependent Changes in Spike Threshold Enhance Feature Selectivity in Rat Barrel Cortex Neurons. *Journal of Neuroscience*, 25(11):2983–2991, mar 2005.

[21] Douglas L Jones, Erik C Johnson, and Rama Ratnam. A stimulus-dependent spike threshold is an optimal neural coder. *Frontiers in Computational Neuroscience*, 9, 2015.

[22] Guo Sheng Yi, Jiang Wang, Kai-Ming Tsang, Xi-Le Wei, and Bin Deng. Input-output relation and energy efficiency in the neuron with different spike threshold dynamics. *Frontiers in Computational Neuroscience*, 9:62, 2015.

[23] Vladimir Itskov, Carina Curto, Eva Pastalkova, and György Buzsa. Cell assembly sequences arising from spike threshold adaptation keep track of time in the hippocampus. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 31(8):2828–2834, feb 2011.

[24] D. A. Henze and G. Buzsaki. Action potential threshold of hippocampal pyramidal cells in vivo is increased by recent spiking activity. *Neuroscience*, 105(1):121–130, 2001.

[25] Rony Azouz and Charles M. Gray. Dynamic Spike Threshold Reveals a Mechanism for Synaptic Coincidence Detection in Cortical Neurons in vivo, 2000.

[26] Adaptive coincidence detection and dynamic gain control in visual cortical neurons in vivo. *Neuron*, 37(3):513–523, feb 2003.

[27] Bertrand Fontaine, José Luis Peña, and Romain Brette. Spike-Threshold Adaptation Predicted by Membrane Potential Dynamics In Vivo. *PLoS Computational Biology*, 10(4):1–11, 2014.

[28] A. Howard MacKenzie and Edwin W. Rubel. Dynamic spike thresholds during synaptic integration preserve and enhance temporal response properties in the avian cochlear nucleus. *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience*, 30(36):12063–12074, sep 2010.

[29] Nicolas Fourcaud-Trocmé, David Hansel, Carl van Vreeswijk, and Nicolas Brunel. How spike generation mechanisms determine the neuronal response to fluctuating inputs. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 23(37):11628–40, dec 2003.

[30] Vladimir Itskov, Carina Curto, Eva Pastalkova, and György Buzsáki. Adaptive Spike Threshold Enables Robust and Temporally Precise Neuronal Encoding. *PLoS computational biology*, 12(6):e1004984, jun 2016.

[31] Bruno A Olshausen and David J Field. Sparse coding of sensory inputs. *Current Opinion in Neurobiology*, 14(4):481–487, aug 2004.

[32] Mark Stopfer. Central processing in the mushroom bodies. *Current Opinion in Insect Science*, 2014.

[33] Bruno A. Olshausen and David J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images, 1996.

[34] Ben D B Willmore, James a Mazer, and Jack L Gallant. Sparse coding in striate and extrastriate visual cortex. *J Neurophysiol*, (April 2011):2907–2919, 2011.

[35] Michael R DeWeese, M. S. Wehr, and A.M. Zador. Binary spiking in auditory cortex. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 23(21):7940–9, 2003.

[36] Tomáš Hromádka, Michael R Deweese, Anthony M Zador, and S Figs. Sparse Representation of Sounds in the Unanesthetized Auditory Cortex. *PLoS Biology*, 6(1):e16, 2008.

[37] John T. Wixted, Larry R. Squire, Yoonhee Jang, Megan H. Papesh, Stephen D. Goldinger, Joel R. Kuhn, Kris A. Smith, David M. Treiman, and Peter N. Steinmetz. Sparse and distributed coding of episodic memory in neurons of the human hippocampus. *Proceedings of the National Academy of Sciences*, 111(26):9621–9626, 2014.

[38] S. Waydo, A. Kraskov, R. Quian Quiroga, I. Fried, and C. Koch. Sparse Representation in the Human Medial Temporal Lobe. *Journal of Neuroscience*, 26(40):10232–10234, 2006.

[39] John T. Wixted, Stephen D. Goldinger, Larry R. Squire, Joel R. Kuhn, Megan H. Papesh, Kris A. Smith, David M. Treiman, and Peter N. Steinmetz. Coding of episodic memory in the human hippocampus. *Proceedings of the National Academy of Sciences*, 115(5):201716443, jan 2018.

[40] Pentti Kanerva. Associative neural memories. chapter Sparse Distributed Memory and Related Models, pages 50–76. Oxford University Press, Inc., New York, NY, USA, 1993.

[41] Ramón Huerta, Thomas Nowotny, Marta García-Sanchez, H. D. I. Abarbanel, and M. I. Rabinovich. Learning Classification in the Olfactory System of Insects. *Neural Computation*, 16(8):1601–1640, aug 2004.

[42] Wolfgang Meyerhof and Sigrun Korsching. *Chemosensory systems in mammals, fishes, and insects.*, volume 47. Springer, 2009.

[43] Eduardo Serrano, Thomas Nowotny, Rafael Levi, Brian H Smith, and Ramón Huerta. Gain Control Network Conditions in Early Sensory Coding. *PLOS Computational Biology*, 9(7):e1003133, jul 2013.

[44] Sigrun Korsching. Olfactory maps and odor images. *Current Opinion in Neurobiology*, 12(4):387–392, 2002.

[45] F. Theunissen and J. P. Miller. Temporal encoding in nervous systems: A rigorous definition. *Journal of Computational Neuroscience*, 2(2):149–162, 1995.

[46] Silke Sachse and C. Giovanni Galizia. Role of Inhibition for Temporal and Spatial Odor Representation in Olfactory Output Neurons: A Calcium Imaging Study. *Journal of Neurophysiology*, 87(2):1106–1117, 2002.

[47] Collins Assisi and Maxim Bazhenov. Synaptic inhibition controls transient oscillatory synchronization in a model of the insect olfactory system. *Frontiers in Neuroengineering*, 5(April):1–10, 2012.

[48] Maxim Bazhenov, Mark Stopfer, Terrence J. Sejnowski, and Gilles Laurent. Fast odor learning improves reliability of odor responses in the locust antennal lobe. *Neuron*, 46(3):483–492, 2005.

[49] Paul Pawletta and Amir Mamlouk. Modeling odor responses of projection neurons and Kenyon cells in insects. *Flavour*, 3(Suppl 1):P13, 2014.

[50] Pavel Sanda, Tiffany Kee, Nitin Gupta, Mark Stopfer, and Maxim Bazhenov. Classification of odorants across layers in locust olfactory pathway. *Journal of Neurophysiology*, 115(5):2303–2316, 2016.

[51] Nitin Gupta and Mark Stopfer. Olfactory coding: Giant inhibitory neuron governs sparse odor codes. *Current Biology*, 21(13):R504–R506, 2011.

[52] Beulah Leitch, Beulah Leitch, Gilles Laurent, and Gilles Laurent. GABAergic Synapses in the Antenna1 Lobe and Mushroom Body of the Locust Olfactory System. *the Journal of Comparative Neurology*, 514(372):487–514, 1996.

[53] Maria Papadopoulou, Stijn Cassenaer, Thomas Nowotny, and Gilles Laurent. Normalization for sparse encoding of odors by a wide-field interneuron. *Science*, 332(6030):721–725, 2011.

[54] Collins Assisi, Mark Stopfer, Gilles Laurent, and Maxim Bazhenov. Adaptive regulation of sparseness by feedforward inhibition. *Nature Neuroscience*, 10(9):1176–1184, 2007.

[55] J. Perez-Orive. Intrinsic and Circuit Properties Favor Coincidence Detection for Decoding Oscillatory Input. *Journal of Neuroscience*, 24(26):6037–6047, 2004.

[56] Joaquín J. Torres and Pablo Varona. Modeling Biological Neural Networks. In Grzegorz Rozenberg, Thomas Bäck, and Joost N. Kok, editors, *Handbook of Natural Computing*. Springer, Berlin Heidelberg, 2012.

[57] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 177(4):500–544, 1952.

[58] J L Hindmarsh and R M Rose. A model of neuronal bursting using three coupled first order differential equations. *Proceedings of the Royal Society of London B: Biological Sciences*, 221(1222):87–102, 1984.

[59] Wulfram Gerstner, Werner M. Kistler, Richard Naud, and Liam Paninski. *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge University Press, 2014.

[60] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[61] A. Destexhe, Z. F. Mainen, and T. J. Sejnowski. An efficient method for computing synaptic conductances based on a kinetic model of receptor binding. *Neural Computation*, 6(1):14–18, Jan 1994.

[62] Shaul Hestrin, Pankaj Sah, and Roger A. Nicoll. Mechanisms generating the time course of dual component excitatory synaptic currents recorded in hippocampal slices. *Neuron*, 5(3):247–253, sep 1990.

[63] C E Jahr and C F Stevens. Voltage dependence of NMDA-activated macroscopic conductances predicted by single-channel kinetics. *The Journal of neuroscience*, 10(9):3178–3182, 1990.

[64] Laurene Fausett, editor. *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1994.

[65] F. B. Rodríguez and R. Huerta. Techniques for temporal detection of neural sensitivity to external stimulation. *Biol Cybern*, pages 289–297, 2009.

[66] M. A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

[67] Sebastian Ruder. An overview of gradient descent optimization algorithms. sep 2016.

[68] Alexander Vergara, Shankar Vembu, Tuba Ayhan, Margaret A. Ryan, Margie L. Homer, and Ramón Huerta. Chemical gas sensor drift compensation using classifier ensembles. *Sensors and Actuators, B: Chemical*, 166-167, 2012.

[69] *OpenCV, Version 3.4.3*, 2018.

[70] Ramón Huerta and Thomas Nowotny. Fast and robust learning by reinforcement signals: explorations in the insect brain. *Neural Computation*, 21(8):2123–2151, aug 2009.

[71] F. B. Rodríguez, R. Huerta, and M. L. Aylwin. Neural sensitivity to odorants in deprived and normal olfactory bulbs. *Plos One*, 8(4):e60745, 2013.

A

Herramientas software desarrolladas

En este anexo se describen las herramientas software que se han desarrollado para implementar el modelo del sistema olfativo de la langosta voladora propuesto en el proyecto. Se presenta una breve descripción de las clases implementadas, sus métodos y su funcionamiento, así como el código fuente de las herramientas.

A.1. Descripción de las herramientas desarrolladas

A continuación se describe cada una de las clases y herramientas desarrolladas con los métodos que contiene. En el código adjunto pueden consultarse todos los detalles acerca de su implementación.

- Datos: esta clase contiene los atributos y métodos para leer los ficheros que contienen los patrones que se van a presentar al modelo y aplicar algunas transformaciones antes de utilizarlos para entrenar la red o clasificarlos. Los métodos que contiene esta clase son:
  - normalizarDatos(): normaliza los atributos de los datos cargados mediante la unidad tipificada (Z-score).
  - normalizarDatosParams: normaliza los atributos de los datos cargados mediante la unidad tipificada dada una media y una desviación típica determinadas que se pasan como parámetros.
  - sesgo(): añade una columna de sesgo a los datos cargados.
  - controlGanancia(): aplica el método sencillo de control de ganancia propuesto a los datos según se describe en la Sección 3.1.
- EstrategiaValidacion: este módulo contiene varias clases sencillas encargadas de organizar las diferentes particiones en las que se van a dividir los datos a clasificar según se vaya a utilizar validación simple o validación cruzada. Las clases que contiene este módulo son:
  - Particion: contiene dos arrays que almacenan los índices de los patrones que se van a usar como train y los índices de los patrones que se van a usar como test.

  - ValidacionSimple: crea un número determinado de particiones de validación simple. Mediante el método *creaParticiones()* selecciona aleatoriamente los índices de los patrones que se usarán en el entrenamiento y los que se usarán como test. La proporción entre la cantidad de patrones de entrenamiento y test es un atributo de la clase. Devuelve un array de objetos Particion con los índices de los patrones de cada tipo.
  - Validación cruzada: crea un número determinado de iteraciones de validación cruzada. Mediante el método *creaParticiones()*, divide el conjunto de datos en el número de particiones seleccionado y designa como patrones de test a una partición diferente para cada iteración. Devuelve un array de objetos Particion con los índices de los patrones de cada tipo.

- RedUmbrales: esta clase contiene todos los métodos y atributos necesarios para el funcionamiento de la red neuronal del modelo del sistema olfativo. Sus atributos son:

  - Nentrada: número de atributos de los patrones del problema de clasificación que se va a resolver.
  - Noutput: número de clases del problema de clasificación.
  - pc: parámetro p_c del sistema que controla la probabilidad de conexión entre una PN y una KC (detalles en la Sección 3.1).
  - s: parámetro s del sistema que controla la actividad en la capa de las KCs (detalles en la Sección 3.1).
  - reg: indica si el término de regulación de actividad (ART, más detalles en Sección 3.1) está o no activo durante el entrenamiento.
  - batchSize: indica el tamaño de batch que se va a usar en el entrenamiento ()
  - momentum: indica el valor que va a tomar el momentum.
  - alpha: indica el valor de la tasa de aprendizaje.
  - errorMin: valor de la función de coste que la red debe alcanzar para detener el entrenamiento.

  Los métodos que contiene esta clase son:

  - entrenar(): este método lleva a cabo la inicialización aleatoria de los pesos y umbrales del modelo y el entrenamiento del mismo presentándole los patrones de entrenamiento agrupados en conjuntos de tamaño batchSize. Para cada patrón, se calcula la actualización de los pesos y umbrales, que se va acumulando. Cuando todos los patrones del grupo han sido presentados, los parámetros de la red se actualizan. Si tras la actualización, el valor de la función de coste se ha incrementado, la actualización se desecha y se restablece el valor anterior de los parámetros. Si el valor de la función de coste es menor que errorMin, entonces el entrenamiento finaliza.
  - clasificar(): clasifica los patrones de test con el modelo previamente entrenado.
  - error(): dadas las clases predichas por el modelo y las clases reales, calcula el porcentaje de error que se ha cometido.
  - feedforward(): método que se encarga de presentar los patrones a la red, calculando el valor de activación de las neuronas de la capa oculta y de las de salida.
  - backpropagationMSE(): contiene las fórmulas para calcular la actualización de los pesos y umbrales en función del error cometido cuando se utiliza el MSE como función de coste.
  - backpropagationCE(): contiene las fórmulas para calcular la actualización de los pesos y umbrales en función del error cometido cuando se utiliza la entropía cruzada como función de coste.

- inicializarPesosOcultas(): inicializa los valores de conexión entre las neuronas de la capa de entrada (PNs) y las de la capa oculta (KCs) dado un determinado p_c.

A.2. Código

En esta sección se presenta el código fuente de las clases presentadas en el apartado anterior.

A.2.1. Datos

```
class Datos(object):
    def __init__(self, fichero, Nclases):
        self.datos = np.genfromtxt(fichero)
        self.Nclases = Nclases

    def sesgo(self):
        sesgo = np.empty(self.datos.shape[0])
        sesgo.fill(1.0)
        self.datos = np.hstack((sesgo[:, np.newaxis], self.datos))

    def normalizarDatos(self):
        means = np.mean(self.datos[:, :-self.Nclases], axis=0)
        stds = np.std(self.datos[:,:-self.Nclases], axis=0)
        stds += 0.000001
        self.datos[:,:-self.Nclases] = (self.datos[:,:-self.Nclases]
            - means)/stds
        return means, stds

    def normalizarDatosParams(self, means, stds):
        self.datos[:,:-self.Nclases] = (self.datos[:,:-self.Nclases]
            - means)/stds

    def controlGanancia(self):
        means = np.mean(self.datos[:, :-self.Nclases], axis = 1)
        stds = np.std(self.datos[:,:-self.Nclases], axis=1)
        stds += 0.000001
        self.datos[:,:-self.Nclases] = ((self.datos[:,:-self.Nclases
            ].T - means)/stds).T
```

A.2.2. EstrategiaParticionado

```
class Particion():
    def __init__(self):
        self.indicesTrain = []
        self.indicesTest = []

class EstrategiaParticionado(object):
    __metaclass__ = ABCMeta
    nombreEstrategia = "null"
    particiones = []
```

```
    @abstractmethod
    def creaParticiones(self, datos, seed=None):
        pass

class ValidacionSimple(EstrategiaParticionado):
    nombreEstrategia="ValidacionSimple"
    def __init__(self, nParticiones, porcentajeTrain=0.5):
        self.porcentajeTrain = porcentajeTrain
        self.nParticiones = nParticiones

    # Crea particiones segun el metodo de validacion simple con una
        determinada proporcion entre train y test
    # Devuelve una lista de particiones
    def creaParticiones(self, datos, seed=None):
        self.particiones = []
        if not seed:
            seed = int(time.time())
        np.random.seed(seed)

        def creaParticion():
            permutacion = np.random.permutation(len(datos))
            particion = Particion()
            nTrain = int(len(datos) * self.porcentajeTrain)
            particion.indicesTrain = permutacion[:nTrain]
            particion.indicesTest = permutacion[nTrain:]
            return particion

        for _ in range(0, self.nParticiones):
            self.particiones.append(creaParticion())

class ValidacionCruzada(EstrategiaParticionado):
    nombreEstrategia="ValidacionCruzada"
    def __init__(self, nFolds):
        self.nFolds = nFolds

    # Crea particiones para validacion cruzada.
    # El conjunto de entrenamiento se crea con las nfolds-1
       particiones
    # y el de test con la particion restante
    # Devuelve una lista de particiones
    def creaParticiones(self, datos, seed=None):
        self.particiones = []
        if not seed:
            seed = int(time.time())
        np.random.seed(seed)
        permutacion = np.random.permutation(len(datos))
        size = len(datos) // self.nFolds

        def creaParticion(i):
            particion = Particion()
            if i == self.nFolds - 1:
```

```
                particion.indicesTest = permutacion[size * i:]
                particion.indicesTrain = permutacion[: size * i]
            else:
                particion.indicesTest = permutacion[size * i: size
                    * (i + 1)]
                particion.indicesTrain = np.append(permutacion[:
                    size * i], permutacion[size * (i + 1):])

            return particion

        for i in range(0, self.nFolds):
            self.particiones.append(creaParticion(i))
```

A.2.3. RedUmbrales

```
class RedUmbrales(object):

    def __init__(self, Nentrada, Noutput, pc, s, batchSize,
        regularizacion=True):
        self.Nentrada = Nentrada
        self.Noutput = Noutput
        self.Nhidden = self.Nentrada * 50
        self.pc = pc
        self.s = s
        self.reg = regularizacion
        self.batchSize = batchSize

    # funcion de transferencia calculada con la funcion expit de
        scipy
    def transferencia(self, num):
        return expit(num)

    # derivada de la funcion de transferencia
    def devTransferencia(self, trans):
        return trans * (1 - trans)

    def feedforward(self, patron):
        # calculo activacion de las neuronas de la capa oculta
        hIn = np.add.reduce(patron * self.pesosOcultas, axis = 1)
        # aplico los umbrales
        hIn = hIn - self.theta
        # calculo la activacion con la funcion de transferencia
        h = self.transferencia(hIn)
        # agrego la neurona que actua como sesgo
        h = np.append(np.array([1.0]), h)
        # calculo la activacion de las neuronas de la capa de salida
        yIn = np.add.reduce(h * self.pesosSalidas, axis=1)
        # aplico los umbrales
        yIn = yIn - self.epsilon
        # calculo su activacion con la funcion de transferencia
        y = self.transferencia(yIn)
```

```
        return y, h, yIn, hIn

    # funcion que calcula el valor del termino de regulacion de
       actividad para la actualizacion de los umbrales
    def regularizacion(self, h):
        primerTerm = 1.0/self.Nhidden * np.sum(h) - self.s
        return primerTerm * self.devTransferencia(h)*(-1)

        def backpropagationMSE(self, y, h, yIn, hIn, patron, clases,
            alpha):
        # actualizacion pesos wjk
        actSalida = (np.repeat((y - clases)*self.devTransferencia(y)[
            None,:], self.Nhidden+1, axis=0) * h[:,None]).reshape(self
            .Noutput, self.Nhidden+1)

        # actualizacion umbrales epsilon
        actEpsilon = (y - clases)*self.devTransferencia(y)*(-1)

        # actualizacion umbrales theta
        actTheta = (np.dot((y - clases)*self.devTransferencia(y),self
            .pesosSalidas)[1:]* self.devTransferencia(h[1:])) * (-1)
        if self.reg:
            actTheta += self.regularizacion(h[1:])

        return actSalida, actEpsilon, actTheta

    def backpropagationCE(self, y, h, yIn, hIn, patron, clases, alpha
        ):
        # actualizacion pesos wij'
        actSalida = (np.repeat((y - clases)[None,:], self.Nhidden+1,
            axis=0) * h[:,None]).reshape(self.Noutput, self.Nhidden+1)

        # actualizacion umbrales epsilonj
        actEpsilon = (y - clases)*(-1)

        # actualizacion umbrales thetak
        actTheta = (np.dot((y - clases), self.pesosSalidas)[1:] *
            self.devTransferencia(h[1:])) * (-1)

        # aniadimos la regularizacion en el caso de que este activa
        if self.reg:
            actTheta += self.regularizacion(h[1:])

        return actSalida, actEpsilon, actTheta

    def inicializarPesosOcultas(self):
            # inicializo a cero todos los pesos entre las PNs y
                las KCs
            self.pesosOcultas = np.zeros((self.Nhidden, self.Nentrada
                ))
            # calculo una matriz de numeros aleatorios de tamanio
```

```
            nPNs x nKCs
        conexiones = np.random.rand(self.Nhidden, self.Nentrada)
        # pongo a uno los valores de los pesos que se
            correspondan con valores mayores que pc en la matriz
            aleatoria
        indices = np.where(conexiones <= self.pc)
        self.pesosOcultas[indices] = 1.0

    def clasificar(self, patrones):
        salida = np.zeros((patrones.shape[0], self.Noutput))
        acts = np.array([])
        nActs = np.array([])
        conteoActivacion = np.zeros(self.Nhidden)

        for i, p in enumerate(patrones):
            # calculo la activacion de las neuronas para un patron
            y, h, _, _ = self.feedforward(p)
            # asigno la clase como la neurona de salida que mas
                activacion tiene
            clase = np.argmax(y)
            y[:] = 0.0
            y[int(clase)] = 1.0
            salida[i, :] = y
            conteoActivacion[h[1:] >= 0.7] += 1.0
            # calculo la activacion media de las KCs para cada patron
                presentado
            acts = np.append(acts, np.mean(h))
            nActs = np.append(nActs, len(np.where(h[1:] >= 0.7)[0]))

        return salida, np.mean(acts), np.mean(nActs), np.std(nActs)

    # calcula el porcentaje de error dadas las predicciones de la red
        y las clases reales
    def error(self, salida, clases):
        errores = salida == clases
        errores = np.logical_and.reduce(errores, axis=1).astype(int)
        erroresCount = np.bincount(errores)

        return float(erroresCount[0])/clases.shape[0], errores

    def entrenar(self, nEpocas, patrones, clases, alpha, errorMin,
        datosTest, clasesTest, momentum):
        # inicializo aleatoriamente los pesos y umbrales
        self.pesosSalidas = np.random.uniform(-0.5, 0.5, (self.
            Noutput, self.Nhidden + 1))
        self.inicializarPesosOcultas()
        self.pesosOcultas = self.pesosOcultas.astype(np.float128)
        self.pesosSalidas = self.pesosSalidas.astype(np.float128)

        self.theta = np.random.rand(self.Nhidden)
        self.epsilon = np.random.rand(self.Noutput)
```

```
            # inicializo los acumuladores de las actualizaciones
                para un batch
        actSalidasBatch = np.zeros((self.Noutput, self.Nhidden + 1))
        actThetaBatch = np.zeros(self.Nhidden)
        actEpsilonBatch = np.zeros(self.Noutput)

        # calculo cuantos batches tengo que presentar durante una
            epoca para mostrar
        # a la red todos los patrones
        epochIterations = int(np.ceil(patrones.shape[0]/self.
            batchSize))

        # inicializo el historico de errores
        errHist = np.array([])
            # comienzo el entrenamiento
        for j in range(nEpocas):
            # copio los valores de los parametros anteriores por
                si la actualizacion hace que la
            # red funcione peor
            copiaSalida = deepcopy(self.pesosSalidas)
            copiaTheta = deepcopy(self.theta)
            copiaEpsilon = deepcopy(self.epsilon)

            errores = 0.0
            # muestro a la red los patrones de cada batch
            for i in range(epochIterations):
                # obtengo los patrones que estan dentro del batch
                    actual
                patronesEpoch = patrones[(i * self.batchSize):(i *
                    self.batchSize + self.batchSize), :]
                clasesEpoch = clases[(i * self.batchSize):(i * self.
                    batchSize + self.batchSize), :]

                # inicializo a cero los acumuladores de
                    actualizaciones
                actSalidasBatch = np.zeros((self.Noutput, self.
                    Nhidden + 1))
                actThetaBatch = np.zeros(self.Nhidden)
                actEpsilonBatch = np.zeros(self.Noutput)

                                # muestro a la red los patrones del
                                    batch uno por uno y calculo la
                                    actualizacion
                for p, c in zip(patronesEpoch, clasesEpoch):
                    y, h, yIn, hIn = self.feedforward(p)
                    actSalida, actEpsilon, actTheta = self.
                        backpropagationCE(y, h, yIn, hIn, p, c, alpha)
                    errores += log_loss(c, y)

                # acumulo las actualizaciones
```

```
                actSalidasBatch += actSalida
                actThetaBatch += actTheta
                actEpsilonBatch += actEpsilon

            # una vez he mostrado todos los patrones, actualizo
                los parametros de la red
             self.pesosSalidas -= actSalidasBatch * alpha + self.
                momentum * actSalidaAnt
             self.theta -= actThetaBatch * alpha + self.momentum *
                 actThetaAnt
             self.epsilon -= actEpsilonBatch * alpha + self.
                momentum * actEpsilonAnt

             # elimino los umbrales negativos, ya que no tienen
                sentido biologico
             self.theta[self.theta < 0.0] = 0.0
             self.epsilon[self.epsilon < 0.0] = 0.0
             # calculo el valor medio de la funcion de coste para
                este batch
             error = 1/(self.Noutput * patronesEpoch.shape[0]) *
                errores
                    # guardo el nuevo error en el historico
        errHist = np.append(errHist, error)

        # si la iteracion es multiplo de
        if error

        # si la nueva actualizacion ha empeorado el rendimiento
           de la red, se restablece el valor
        # de los parametros al anterior
        if j > 2 and (errHist[-2] - errorEnt) < 0.0:
             self.pesosSalidas = copiaSalida
             self.theta = copiaTheta
             self.epsilon = copiaEpsilon

    return error, j
```