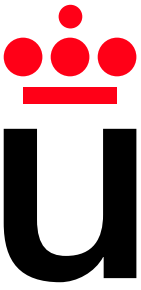


Universidad
Rey Juan Carlos

Escuela Técnica Superior
de Ingeniería Informática

Grado en Ingeniería del Software

Curso 2023-2024

Trabajo Fin de Grado

MÉTODOS DE APRENDIZAJE AUTOMÁTICO PARA LA CLASIFICACIÓN DE COMPORTAMIENTOS ESTEREOTIPADOS

Autor: Gonzalo Ortega Carpintero
Tutor académico: Alberto Fernández Gil
Tutora CSIC: Paula Peixoto Moledo

Agradecimientos

(Por redactar)

Resumen

(Por redactar)

Palabras clave:

- Python
- Neurociencia
- Aprendizaje automático
- ...

Índice de contenidos

Índice de tablas

Índice de figuras

Índice de códigos

1 Introducción

Este documento recoge el Trabajo de Fin de Grado (TFG) de Gonzalo Ortega Carpintero, incluido en el itinerario del grado en Ingeniería del *Software* de la Universidad Rey Juan Carlos.

En este primer capítulo se presenta la motivación que ha dado lugar a la idea y desarrollo de este TFG en la Sección 1.1. En la Sección 1.2 se introduce el objetivo principal del trabajo y se enumeran los objetivos parciales que se han propuesto para conseguirlo. En la Sección 1.3 se hace un repaso a la metodología seguida para realizar el trabajo y finalmente, en la Sección 1.4 se introducen las herramientas y bibliotecas que se han utilizado.

1.1. Motivación

La Neurociencia es la disciplina científica que estudia el sistema nervioso y todas sus componentes. Para el estudio del funcionamiento del cerebro pueden utilizarse multitud de técnicas diferentes, pero para evitar la intrusión en el sujeto de estudio de muchas de ellas, muchos experimentos e investigaciones se basan en el estudio del comportamiento. Es la etología la encargada de este tipo de estudio en animales y en la cual, históricamente, la categorización de comportamientos dependía fuertemente de la observación manual. Eso ha conllevado siempre una excesiva cantidad de tiempo y las desventajas de los posibles errores de percepción humanos, limitando la cantidad y la calidad de los análisis a realizar.

Gracias a los recientes avances en algoritmos de aprendizaje automático, hoy en día es posible computar multitud de datos de forma simultánea, agilizando y automatizando análisis como el ilustrado previamente.

Este Trabajo de Fin de Grado se ha realizado a lo largo de una estancia de prácticas académicas en el Jercog's Team, un equipo de investigación perteneciente al Instituto Cajal, instituto de Neurociencia del Congreso Superior de Investigaciones Científicas. El equipo está centrado en el estudio de la memoria mediante experimentos con ratones, y uno de sus proyectos abiertos consistía en poder elaborar una herramienta para clasificar automáticamente los comportamientos estereotipados de los ratones. Estos son comportamientos cortos, repetitivos y con cierta tendencia a generar patrones, tales como rascarse, caminar en círculos o levantarse a dos patas, ejecutados sin ninguna finalidad, y usualmente inducidos por estar en entornos cerrados y artificiales.

Para la ejecución de uno de sus últimos experimentos utilizan ratones tratados con dimetilnitrosamina (NMDA de sus siglas en inglés, *N-Nitrosodimethylamine*), un compuesto que bloquea la capacidad de los ratones para almacenar memoria a largo plazo. En las sesiones de experimentación los ratones tratados ejercen las mismas tareas que ratones control, tratados con un suero placebo, y en las grabaciones hay ratones de ambos tipos. A ojo de un experimentador, es complejo determinar a ciegas si un ratón está bajo los efectos de NMDA o es control, por lo que es interesante la posibililidad de contar con una herramienta que sea capaz de distinguir unos de otros a partir de los videos de las sesiones.

Durante la estancia se han valorado multitud de técnicas de análisis y procesado de datos, haciendo hincapié en métodos de aprendizaje automático para tratar de completar el proyecto gracias a los últimos avances en computación.

1.2. Objetivos

El objetivo principal de este trabajo es utilizar diferentes métodos de aprendizaje automático para clasificar de forma automática comportamientos de animales. Para ello, se han planteado los siguientes objetivos parciales:

- Realizar una introducción a técnicas de aprendizaje automático aplicables al procesado de datos en el análisis de experimentos con animales.
- Preparar y preprocesar datos provenientes de archivos de video para su posterior análisis.
- Aplicar distintos algoritmos de agrupación para tratar de separar fragmentos de video y clasificarlos en función del comportamiento que esté realizando el animal.

- Tratar de distinguir de forma automática, mediante la clasificación de los comportamientos o de forma directa, si un animal está bajo los efectos de MDMA.

1.3. Metodología seguida

El trabajo se divide en dos capítulos principales, en el Capítulo 2 abordaremos el estudio teórico de ciertos algoritmos de agrupamiento, reducción de dimensionalidad y de clasificación supervisada. Todo ello desde un punto de vista teórico con ejemplos fáciles de visualizar para ayudar al lector a comprender en profundidad los métodos.

En el Capítulo 3, desarrollaremos paso a paso el análisis real ejercido durante la estancia en el Instituto Cajal. El principal objetivo durante estos meses ha sido buscar la mejor forma de, dados una serie de videos de ratones en su caja, detectar automáticamente diferentes tipos de comportamientos. El fin último del proyecto es estudiar la posibilidad de clasificar dos grupos de animales, control y medicados, basándose en los comportamientos estereotipados.

Para el preprocesado de los videos se ha utilizado la herramienta DeepLabCut, que mediante el uso de redes neuronales y una relativamente pequeña muestra de entrenamiento computa una estimación de la posición de los animales en cada uno de los fotogramas. Tras el preprocesado haremos una revisión de los artículos científicos que han motivado las diferentes técnicas de agrupamiento que hemos acabado utilizando, y finalizaremos desarrollando cada una de ellas a la par que desglosando el código en Python utilizado en cada apartado.

1.4. Bibliotecas y herramientas utilizadas

DeepLabCut

DeepLabCut [1] es una herramienta para la estimación 2D y 3D sin marcadores mediante el uso de redes neuronales. Es capaz de identificar de rastrear diferentes partes del cuerpo de múltiples especies realizando todo tipo de comportamientos. Esta ha sido la base de todo nuestro trabajo, ya que todos los videos que hemos analizado han sido procesados en primer lugar por DeepLabCut para rastrear las posiciones de múltiples puntos de los animales a lo largo de los videos de las sesiones. Debido a su importancia, damos una explicación más en detalle de su funcionamiento y de como lo hemos utilizado en la sección 3.1.1.

Google Colab

Google Colab es una herramienta para realizar cuadernos de Jupyter en línea, y poder ejecutarlos en el *backend* de Google. Estos son documentos que intercalan fragmentos de texto con fragmentos de código ejecutable en Python, así como la salida de las distintas ejecuciones. Todo el codigo realizado para este trabajo ha sido realizado en cuadernos de Colab, y puede ser consultado en los siguientes enlaces:

Ejemplos teóricos	https://colab.research.google.com/drive/1qL-LQTCFLZcqExN7qyYyRAT1nm7P8NhuK?usp=sharing
DeepCutLab	https://colab.research.google.com/drive/1dFb-mUcfW9el50v0RBCkLIItTD6SF7A3x?usp=sharing
Análisis principal	https://colab.research.google.com/drive/1ak2VpDi-zTnV-uEDp-viEkpa8GFDGuBR?usp=sharing

Tabla 1.1: Enlaces a cuadernos de Google Colab.

Scikit-learn

Scikit-learn [2] es una biblioteca de código abierto de Python que contiene numerosas implementaciones de algoritmos de aprendizaje automático. Hemos usado dichas implementaciones tanto como para la explicación teórica de los algoritmos, como para el análisis real de los datos de los animales.

pandas

`pandas` es una biblioteca de manejo de datos mediante tablas denominadas `DataFrame`. Todos los datos que hemos cargado para ser analizados los hemos guardado como DataFrames para poder tener un fácil acceso a todos ellos y a sus variables pudiendo, además, visualizarlos de una forma sencilla.

NumPy

NumPy es una de las bibliotecas principales de cálculo científico en Python. Proporciona un objeto de `array` multidimensional y numerosas funciones estadísticas y algebraicas de mucha utilidad.

matplotlib

`matplotlib` es una biblioteca para crear figuras en Python. Todas las figuras de representación de datos que aparecen en este trabajo han sido creadas con `matplotlib`. El código concreto utilizado para generarlas puede consultarse en los cuadernos de Colab.

2

Fundamentos teóricos

En este Capítulo vamos a dar una introducción varios algoritmos de aprendizaje automático.

2.1. Métodos no supervisados

Los algoritmos de aprendizaje automático no supervisados se utilizan cuando no se conoce la salida esperada. Al algoritmo de aprendizaje se le otorgan los datos de entrada y se le pide extraer información de estos datos. Las principales aplicaciones de estos algoritmos, las cuales vamos a aprovechar, son la agrupación de datos y la reducción de dimensionalidad de las variables de los mismos. Esa última es usada principalmente para poder hacer representaciones de datos multidimensionales, los cuales serian complejos de visualizar de otra forma.

La principal pega que pueden tener estos algoritmos es que, si bien no siempre son capaces de identificar conocimiento dados los datos utilizados, cuando lo obtienen, no siempre es el conocimiento que esperábamos obtener. Póngase el ejemplo de un algoritmo que tratase de agrupar rostros de personas iguales. Al no darle a priori ningún tipo de salida de ejemplo, el algoritmo puede acabar clasificando si los rostros están de frente o de lado, no precisamente lo que esperábamos. Es por ello que estos algoritmos cuentan con diversidad de parámetros para ajustarlos a nuestras necesidades, tratando de realizar la agrupación deseada.

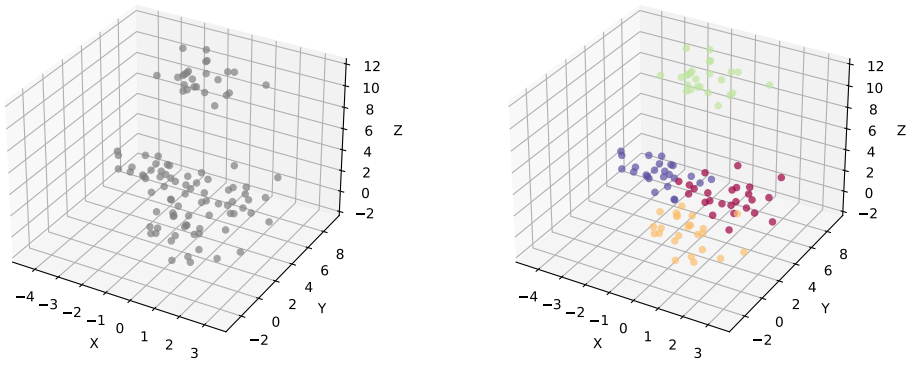
En esta sección vamos a estudiar a fondo tres tipos de algoritmos de agru-

pación: el agrupamiento por k-medias, la agrupación aglomerada, y la agrupación por afinidad. Además, estudiaremos también el principal algoritmo de reducción de dimensiones, el análisis de componentes principales, PCA, de sus siglas en inglés. Los principales ejemplos y explicaciones de los algoritmos han sido inspirados por los dados en el libro [3, Introduction to Machine Learning with Python].

Código 2.1: Generar datos artificiales de prueba.

```
from sklearn.datasets import make_blobs

X, y = make_blobs(n_samples=100, centers=4,
                  n_features=3, random_state=0)
```



(a) (b)

Figura 2.1: Datos artificiales generados para probar algoritmos de agrupamiento. Consisten en 100 puntos tridimensionales agrupados equilibradamente al rededor de 4 centros aleatorios. 2.1a En gris los puntos generados sin asignar ninguna etiqueta. Estos serán los datos que recibirán los algoritmos para procesar y agrupar. 2.1b Datos etiquetados según el grupo al que pertenecen. Esto nos ayudará a verificar la eficacia de nuestros algoritmos.

En la tabla 1.1 está el enlace para poder ejecutar el código en el *backend* de Google.

2.1.1. Análisis de componentes principales

(EXPLICACIÓN PCA)

Código 2.2: Escalar y estandarizar los datos.

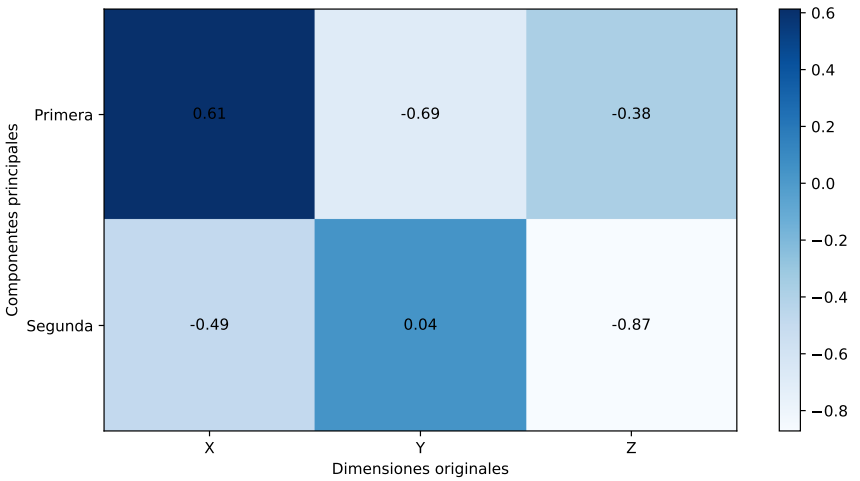
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(X)
features_scaled = scaler.transform(X)
```

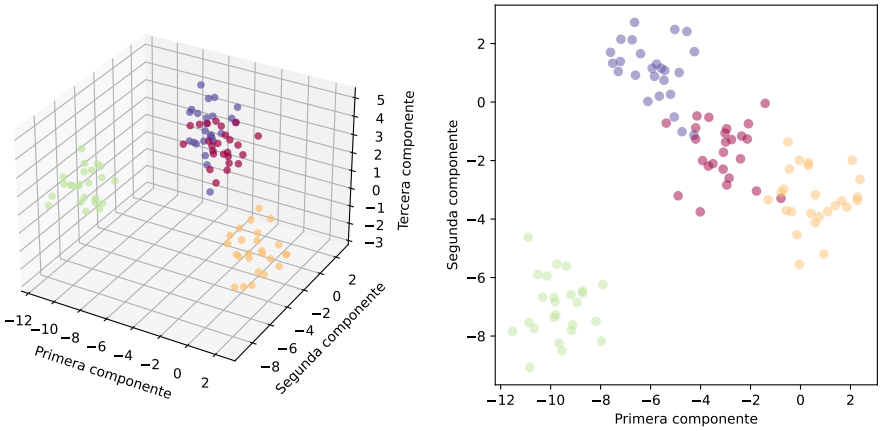
Código 2.3: Realizar un PCA.

```
from sklearn.preprocessing import StandardScaler

pca = PCA(n_components=3)
pca.fit(features_scaled)
X_reduced = pca.transform(X)
```



(a)



(b) (c)

Figura 2.2: 2.2a Mapa de calor con las ponderaciones asignadas por el PCA a cada uno de las dimensiones originales. 2.2b Visualización de los datos previamente generados tras escalarlos y aplicarles la PCA. Los colores de los grupos se mantienen, pero los ejes ahora representan las componentes principales, en vez de las dimensiones originales. 2.2c Visualización de las dos primeras componentes principales sobre el plano. Este será el formato en el que visualizaremos más adelante las agrupaciones ejercidas sobre datos reales de dimensionalidad alta.

2.1.2. Agrupamiento por K-medias

El algoritmo de k-medias clasifica los datos separando los datos en n grupos de con la misma varianza.

El algoritmo comienza inicializando aleatoriamente n centroides, siendo n el número de agrupaciones que se le han dicho que realice. Estos centroides serán los centros de las agrupaciones que va a realizar, no tienen por qué pertenecer a los datos, pero sí que están contenidos en su misma dimensión. En cada iteración, el algoritmo asigna a cada punto de los datos el centroide más próximo y luego asigna a cada agrupación un nuevo centroide calculado como la media de los datos que se han asignado a dicha agrupación.

Formalmente, el algoritmo divide un conjunto de n puntos x en k agrupaciones disjuntas C, cada una descrita por la media μ_j de los puntos en la agrupación. Para ello, el algoritmo trata de encontrar los centroides que minimicen

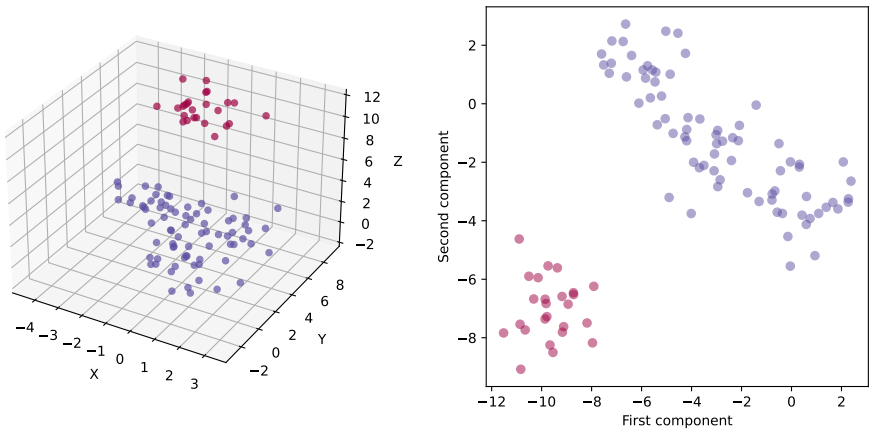
$$\sum_{i=0}^{n} \min_{\mu_j \in C}(||x_i - \mu_j||^2) \tag{2.1}$$

El algoritmo finaliza cuando una iteración no realice ninguna modificación de las agrupaciones.

Código 2.4: k-medias.

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=2, n_init="auto"
kmeans_assigment = kmeans.fit_predict(X)
```



(a) (b)

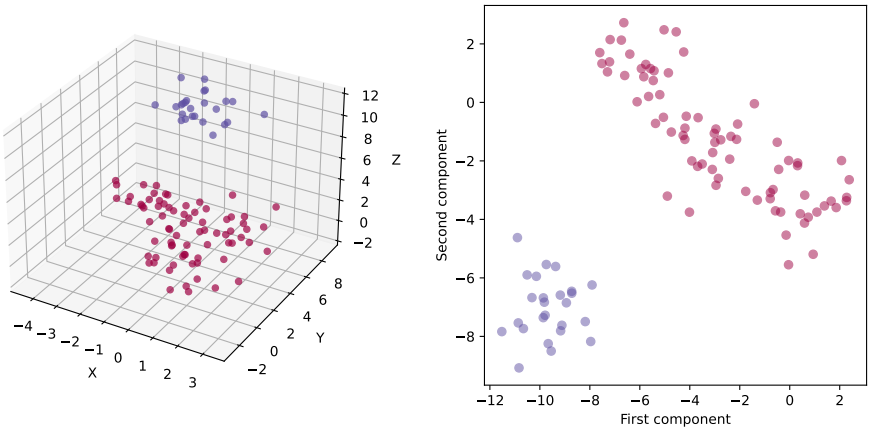
Figura 2.3: 2.3a Visualización sobre los datos sin modificar de los grupos que ha realizado el algoritmo de k-medias. 2.3b Visualización de los grupos realizados sobre los datos procesados por el PCA.

2.1.3. Agrupación aglomerada

(EXPLICACIÓN ALGORITMO)

Código 2.5: Agrupación aglomerada.

```
1  from sklearn.cluster import AgglomerativeClustering
2
3  agg = AgglomerativeClustering()
4  agg_assigment = agg.fit_predict(X)
```



(a) (b)

Figura 2.4: (EXPLICACIÓN FIGURA)

2.1.4. Agrupación por afinidad

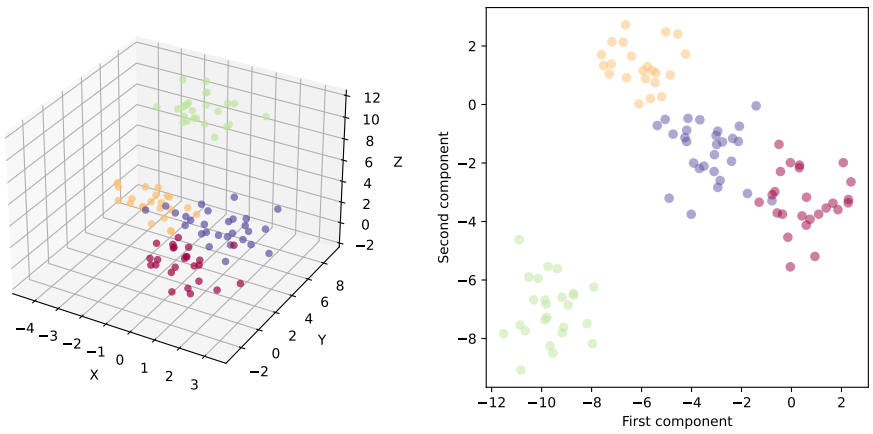
El algoritmo de propagación de afinidad crea agrupaciones mandando mensajes entre pares de puntos hasta que converge. La principal cualidad de este algoritmo es que, a diferencia con la mayoría de algoritmos de agrupamiento, no necesita saber el número de agrupaciones a realizar de antemano, sino que las genera dinámicamente.

(TERMINAR EXPLICACIÓN ALGORITMO)

Código 2.6: Propagación de afinidad.

```
from sklearn.cluster import AffinityPropagation

aff = AffinityPropagation()
aff_assigment = aff.fit_predict(X)
```



(a) (b)

Figura 2.5: (EXPLICACIÓN FIGURA)

2.2. Métodos supervisados

(POR HACER)

[4, Deep Learning with PyTorch]

2.2.1. Red neuronal

3

Análisis y experimentación

Una vez adquiridos los conocimientos teóricos necesarios, y comprendido el funcionamiento de los diferentes algoritmos que vamos a utilizar, estamos en disposición de aplicarlos a los datos de experimentación y comprobar la eficacia de los algoritmos en un caso real. Este capítulo aborda el trabajo realizado durante tres meses de prácticas en el Instituto Cajal, aplicando diferentes técnicas para tratar de agrupar los comportamientos de los animales, y de diferenciar animales bajo efectos de tratamiento o de placebo.

En la Sección 3.1 tratamos como hemos preprocesado los videos para homogeneizar todas las sesiones de experimentación, obtener los datos de posición de los animales en ellas y filtrar e interpolar los datos poco verosímiles. En la Sección 3.2 desarrollamos el proceso de análisis de los datos preprocesados aplicando los diferentes algoritmos vistos anteriormente y analizando sus resultados.

3.1. Preprocesado de datos

Los datos de los que hemos partido para realizar el análisis han sido 88 pares de videos en blanco y negro, de unos cinco minutos cada uno, de ratones en su caja sin estar realizando ninguna tarea concreta. Cada par de videos consistía en un video de la vista cenital de la caja y otro de la vista lateral, como se puede observar en la Figura 3.1. Además, de cada par de videos teníamos el código del animal que estaba siendo grabado, la fecha de la sesión y si el animal estaba bajo los efectos del tratamiento de NDMA en dicha fecha o no. Mostramos un resumen

de estos datos en la Tabla 3.1.

Video	Código del animal	Fecha de la sesión	Tratamiento
1	4128	2020-12-02	CONTROL
2	4128	2020-11-21	CONTROL
3	4128	2020-11-23	CONTROL
...	...	...	...
80	4108	2020-09-25	NMDA
81	4089	2020-09-18	NMDA
82	4089	2020-09-12	CONTROL
83	4089	2020-09-24	NMDA

Tabla 3.1: Fragmento del `DataFrame` que almacenaba los datos de las sesiones, incluyendo el número del video según se almacenaban en memoria, el código del animal, la fecha de la sesión y el estado del tratamiento.

Para tratar de averiguar de forma automática el tipo de tratamiento de cada sesión, podríamos entrenar una red neuronal utilizando directamente los videos como datos de entrenamiento. Sin embargo, no contamos con una cantidad muy elevada de videos para entrenar, y además esto no nos daría ninguna pista sobre como comportamientos concretos se relacionan con el tratamiento. Al no contar con una base de datos de fragmentos de videos clasificados según el comportamiento del animal, tampoco podemos entrenar una red para distinguir estos comportamientos sin tener que pasar antes por un seguramente tedioso periodo de etiquetado manual de fragmentos. Por ello, hemos hecho uso de la herramienta DeepLabCut para rastrear la posición del ratón en cada fotograma y así poder hacer cálculos sobre ella, y trabajar con otros algoritmos de agrupación no supervisada para tratar de clasificar diferentes movimientos del ratón a lo largo del video.

3.1.1. DeepLabCut

(EXPLICACIÓN DEEPLABCUT)

3.1.2. Filtrado e interpolación

Una vez hemos pasado todos los videos por DeepLabCut, tenemos dos ficheros de datos para cada uno de las sesiones, uno para las coordenadas de los puntos de la vista cenital y otro para las coordenadas de los puntos de la vista lateral. Al cargar los ficheros a memoria como `DataFrames` de `pandas`, vemos que no todos ellos tienen la misma duración, por lo que los homogeneizamos todos eliminando los datos de los últimos fotogramas, consiguiendo que cada `DataFrame` tenga un

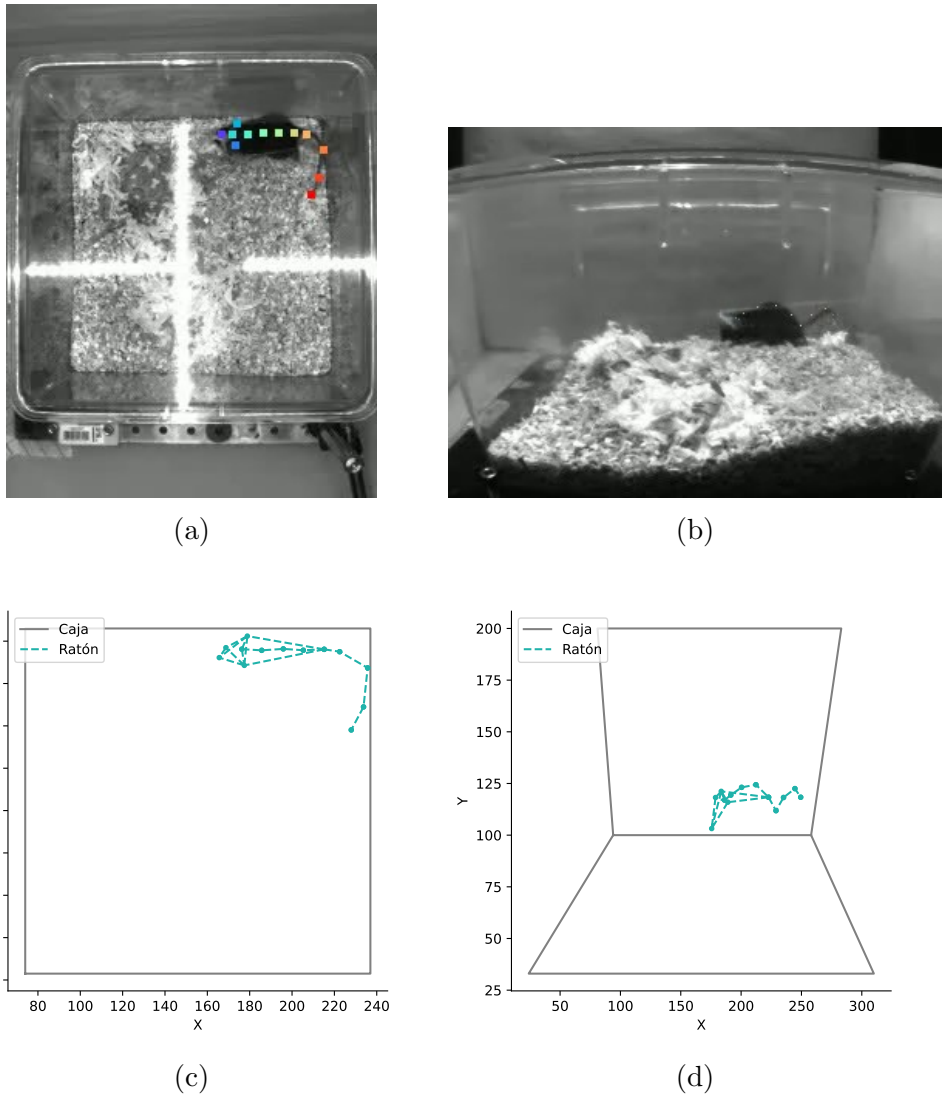


Figura 3.1: Salida de DeepLabCut del animal 4128 el 02-12-2020, 1:00:37. 3.1a Video de la vista cenital de la caja. Los puntos sobre el animal son los dibujados por DeepLabCut para rastrear las partes del animal. 3.1b Video de la vista lateral de la caja. 3.1c En azul, la triangulación formada por los puntos obtenidos por DeepLabCut de la vista cenital. Cinco puntos para la cabeza, cuatro para la espalda y otros cuatro para la cola. En gris, las dimensiones de la base de la caja, obtenidas midiendo las distancias en píxeles sobre un fotograma de video. 3.1d Análogo a la vista cenital, desde la vista lateral. En gris, la altura mínima del suelo de la caja y su pared trasera.

	Nosex	Nosey	Noselikelihood	Headx	Heady	...
0	136.165344	177.722496	0.000084	129.790253	174.772552	...
1	162.032005	201.444756	0.942181	168.152061	202.420639	...
2	156.297043	200.326378	0.000073	162.436028	203.156837	...
3	155.370415	199.043297	0.000277	159.507599	200.199928	...
4	149.272644	197.677170	0.000045	155.493912	198.814835	...
...	...	...	...	...	...	...

Tabla 3.2: Extracto del `DataFrame` de `pandas` de los datos sin procesar de DeepLabCut. Para cada punto rastreado, y para cada fotograma del video, se computa la predicción de la coordenada x y de la coordenada y y se da la verosimilitud de dicha predicción.

tamaño de 6000 filas, lo que equivale a 6000 fotogramas o 5 minutos de video. Además, eliminamos los datos de 4 videos que no llegan a esa duración, ya que son videos con problemas que no entorpecerían el análisis. Uno de ellos solo duraba 3 minutos ya estaba mal grabado y al final estaba a cámara rápida, por ejemplo.

3.1.3. Computo de variables a analizar

Una vez todos los datos han sido preprocessados, podemos computar diferentes propiedades de los mismos para usarlas posteriormente en los algoritmos de agrupación o de reducción de dimensiones. Además, podemos computar puntos de mayor confianza calculando medias de otros puntos individuales.

(FINALIZAR SECCIÓN e INCLUIR FIGURAS)

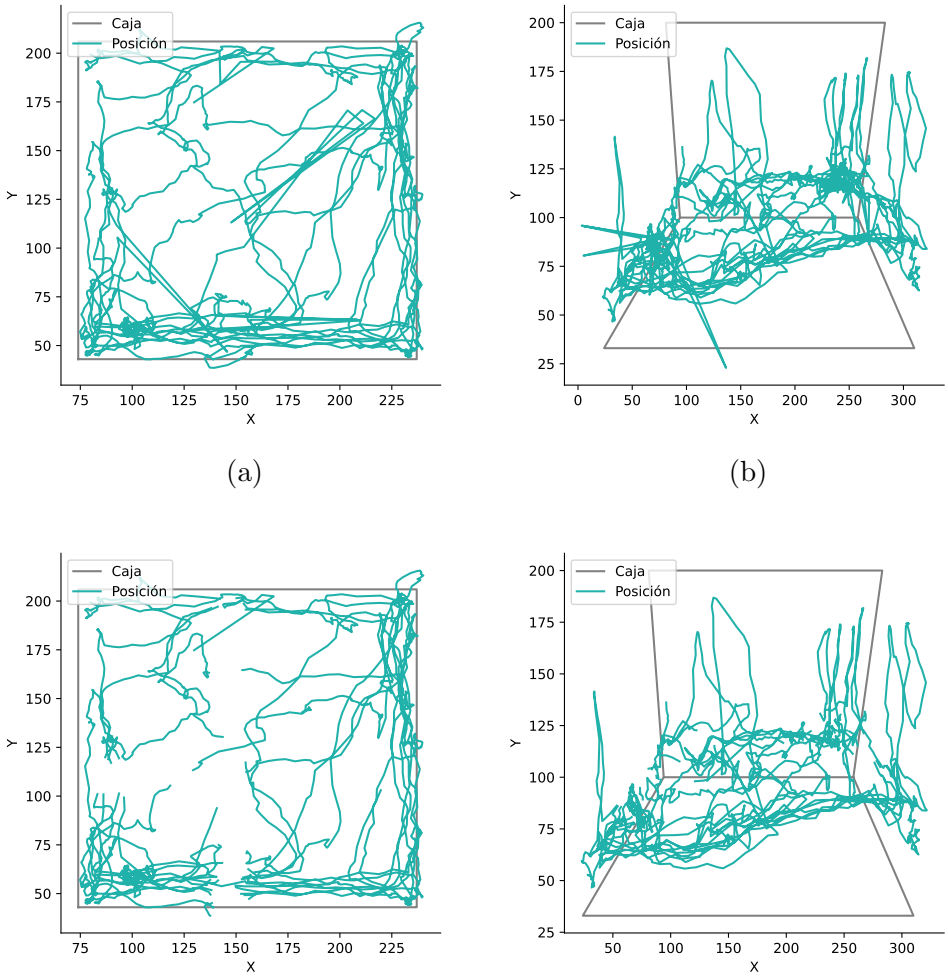
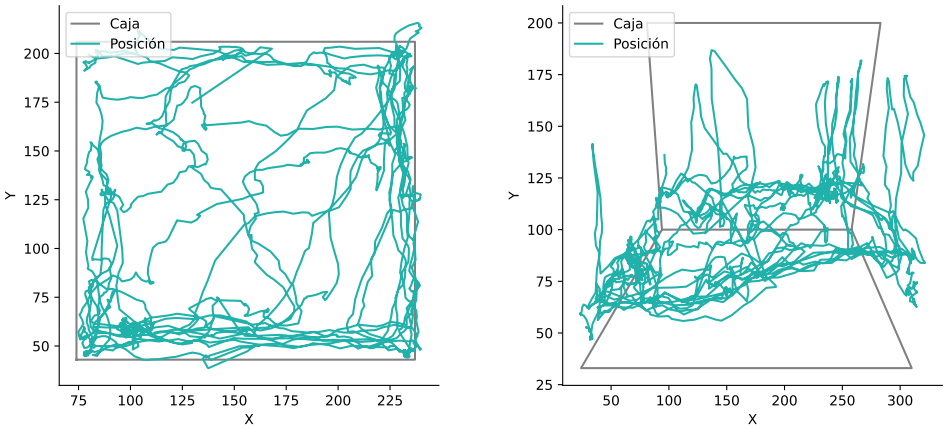


Figura 3.2: Vistas cenital y lateral de trayectorias de un punto de la cabeza del animal. Datos correspondientes a los dos primeros minutos de la sesión del 2020-12-02 del animal 4128. 3.2b: Vista cenital de los datos antes del preprocesado. En gris se ha dibujado las dimensiones de la caja y en color la trayectoria del punto `Head`.



(a) (b)

Figura 3.3: Vistas cenital y lateral de trayectorias de un punto de la cabeza del animal. Datos correspondientes a los dos primeros minutos de la sesión del 2020-12-02 del animal 4128. 3.3a: Vista cenital de los datos filtrados e interpolados. En gris se ha dibujado las dimensiones de la caja y en color la trayectoria del punto `Head`.

3.2. Análisis de datos

3.2.1. Agrupaciones respecto a la trayectoria.

(Explicar como se han fragmentado los videos para tratar de realizar agrupaciones de que está haciendo el animal en cada momento y para tratar de si se trata de un animal bajo los efectos del tratamiento o no.)

3.2.2. Agrupaciones incluyendo más variables computadas.

(Mismo proceso que en la sección anterior pero ahora incluyendo mas variables además de la trayectoria, como velocidades, aceleración, angúlos...)

3.2.3. Agrupamiento mediante una red neuronal.

(Mismo procedimiento, pero ahora tratar de separar los dos tipos de animales entrenando una red neuronal.)

4

Resultados

(Por redactar)

5

Conclusiones

(Por redactar)

Bibliografía

[1] A. Mathis, P. Mamidanna, K. M. Cury, T. Abe, V. N. Murthy, M. W. Mathis, and M. Bethge, "Deeplabcut: markerless pose estimation of user-defined body parts with deep learning," *Nature Neuroscience*, 2018. [Online]. Available: https://doi.org/10.1038/s41593-018-0209-y

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[3] A. C. Müller and S. Guido, *Introduction to Machine Learning with Python.* O'Reilly, 2017.

[4] E. Stevens, L. Antiga, and T. Viehmann, *Deep Learning with PyTorch.* Manning, 2020.