Desarrollo de sistema para calibrar electromiogramas de forma automática

D. Anchuela Cantarero

Ingeniería Biomédica, Universidad San Pablo CEU, Madrid, España, d.anchuela@usp.ceu.es

**Resumen**

El campo del aprendizaje automático facilita el hecho de poder adaptar automáticamente dispositivos médicos a las necesidades de cada paciente. Además, las señales de electromiograma, son muy utilizadas en el control de dispositivos médicos. Este documento explica el proceso de diseño y creación de un sistema para la calibración de electromiogramas de forma automática*.* El estudio se divide en tres partes principales: la primera es el procesado de la señal de EMG recogida a partir de una placa *Bitalino*, para obtener una muestra simplificada de la señal original. La segunda y en la que el estudio se va a centrar más, es en la comparación de diferentes herramientas para el preprocesamiento de los datos, así como de métodos de agrupamiento (o *clustering)* para poder elegir la opción más optima, utilizando para ello el *Davis* *Boulding* *Score*. Por último, se implementará el calibrador en una interfaz paciente-ordenador para probar su eficacia. En este caso se ha utilizado un controlador pensado para el juego *T-Rex* de *Chrome,* en el cual se hará saltar al dinosaurio en el momento en el que se detecte una contracción. Finalmente, las conclusiones del estudio han resultado ser que los métodos de *clustering*, a diferencia de los métodos de preprocesamiento, parecen no tener demasiada influencia en el rendimiento del dispositivo.

1. **Introducción.**

El reciente avance que ha tenido el aprendizaje automático en los últimos años abre las puertas a poder enfocar de una manera más eficiente problemas que anteriormente resultaban más complejos. Uno de los principales campos dónde se puede ver el potencial de este tipo de tecnología es en la medicina. El aprendizaje automático juega un papel fundamental a la hora de realizar toma de decisiones [1], reconocer patrones en diferentes pacientes, predecir posibles enfermedades [2], etc. Esto se logra gracias a la facilidad que tienen los ordenadores para procesar miles de datos en poco tiempo.

Por otra parte, la señal de electromiograma (EMG) es una señal muy usada para controlar diversos dispositivos, lo que la convierte en una de las señales más utilizadas en el diseño de interfaces persona-ordenador[3]. Aunque existen soluciones generales para el control de dispositivos mediante EMG, los controladores más precisos requieren adaptar su funcionamiento a las características individuales del EMG de cada persona. El uso de la inteligencia artificial puede acelerar el desarrollo de dichos controladores.

En este estudio, aprovecharemos las técnicas que nos da este campo para diseñar un calibrador automático aplicable a cualquier dispositivo que se tenga que controlar a partir de una señal de electromiograma. La aplicación del aprendizaje automático en este caso está en que el programa recogerá los datos de la señal de EMG del paciente para posteriormente ajustar los parámetros del controlador de manera que cualquier persona pueda jugar sin que se tenga que ajustar los parámetros manualmente, lo cual es bueno ya que aunque se desee hacer manualmente, nunca se podría producir un resultado tan bueno como el que conseguirá el programa después de entrenarse con los datos de ese jugador. Como parte del diseño del calibrador, se han comparado distintos métodos de preprocesado de electromiografía, así como distintos algoritmos de clustering. Finalmente, se ha implementado la mejor solución en una interfaz que controla un juego

Como prueba de concepto, se ha aplicado el calibrador al famoso juego del *T-Rex* de *Google Chrome*.

El resto del documento se estructura de forma similar a la hoja de ruta empleada durante el proyecto. En primer lugar, para diseñar y validar el sistema de calibración, se recopilaron y almacenaron señales de EMG (sección 2.1). A continuación, se extrajeron de los EMGs las características más relevantes para la detección de contracciones (sección 3.1.1), y se preprocesaron los datos para mejorar el rendimiento de los algoritmos de aprendizaje automático (sección 2.2). Seguidamente se comparó el funcionamiento de distintas combinaciones de preprocesado y de algoritmos de clustering (sección 3.2.2). El calibrador finalmente elegido se validó mediante su uso en el control de un videojuego. (sección 3.3.2).

1. **Materiales.**
   1. En primer lugar, se realizará una recolección de los datos de EMG (sección 2.1). Posteriormente, se hará una revisión de los métodos de *clustering* que se utilizarán para llevar a cabo el estudio (sección 2.2).**Recolección de datos**

Para la recolección de datos hemos utilizado una placa *Bitalino* (sección 2.1.1) siguiendo el procedimiento experimental descrito en la sección 2.1.2.

* + 1. **Placa *Bitalino***

La placa *Bitalino*[4]es un dispositivo ampliamente utilizado a la hora de la recogida de distintos tipos de parámetros fisiológicos debido a su gran versatilidad y a su facilidad de uso.

Las muestras de EMG las recogemos utilizando el software *Open Signals* [5], una placa *Bitalino,* y tres electrodos. Para ello conectamos por Bluetooth el *Bitalino* al ordenador. Posteriormente lo configuramos desde el *Open* *Signals,* conectamos los electrodos a la terminal de EMG de la placa, y nos los ponemos en el brazo de la siguiente manera:

* Electrodos positivo y negativo conectados en dos puntos del antebrazo.
* Electrodo neutro haciendo contacto con una zona del cuerpo sin actividad eléctrica.
  + 1. **Procedimiento con voluntarios**

Los perfiles de los voluntarios son los siguientes:

* Voluntario 1: Varón de 21 años, 82 kg, 1.8m.
* Voluntario 2: Varón de 51 años, 86 kg, 1.77m.
* Voluntario 3: Varón de 17 años, 75 kg, 1.80m.
* Voluntario 4: Mujer de 17 años, 70kg, 1.7m.

Grabamos una muestra de actividad mioelectrica, y la guardamos como *.txt* para analizarla posteriormente.

* 1. **Revisión de clustering**

Una vez tenemos ya nuestros datos separados, necesitamos emplear nuestro modelo de *clustering*, que será el que determinará los diferentes grupos en los que se detecta que se separan nuestros datos.

Entre los algoritmos de clustering existentes, nos centramos en aquellos que permiten especificar el número de *clusters* deseados. Esto se debe a que el objetivo del sistema es discernir entre dos posibles estados: músculo contraído y músculo relajado, por lo que necesitaremos un total de 2 *clusters*. Los algoritmos que incluyen esta posibilidad son los siguientes: :

* *KMeans*
* *Gaussian Mixture Model*
* Agrupamiento jerárquico
  + 1. ***K-Means***

El algoritmo K-Means [6] selecciona aleatoriamente k puntos como centroides, donde k es el número de *clusters* que necesitamos.. Posteriormente evalúa los demás puntos para asignarlos a un grupo, determinando la distancia más cercana que hay entre ese punto en concreto con cada uno de los *clusters.*

La selección de los *clusters* iniciales es aleatoria, pero la asignación de clusters se repite varias veces hasta que dos iteraciones sucesivas no alteren la asignación, lo que significará que el algoritmo ha encontrado una solución.Para minimizar el impacto de la selección aleatoria de los clústers iniciales, el algoritmo se corre varias veces.

* + 1. ***Gaussian Mixture model***

*Modelos de mezcla gaussiana* [7] es un tipo de algoritmo es similar a *KMeans* pero ofrece un poco más de flexibilidad ya que estamos suponiendo que los datos no se distribuyen necesariamente de forma circular. El algoritmo *KMeans* no permite que su matriz de covarianza sea elíptica, sino circular, por lo que en este caso se ve un poco más limitado. Esto se traduce en que en este caso podemos tener *clusters* de forma elíptica. En este caso, a diferencia del anterior, se va a calcular la probabilidad que tiene cada punto de pertenecer a un *cluster* o a otro. Cuanto más cerca esté el punto del centroide del *cluster*, más probable será que pertenezca a él.

* + 1. **Agrupamiento jerárquico**

En el caso del algoritmo de agrupamiento jerárquico [8], el modelo trata a cada dato como un *cluster* diferente. Posteriormente se van fusionando los *clusters* que están más cerca entre sí, hasta que llegamos al número de *clusters* deseados. Para determinar la distancia entre *clusters* se puede hacer con distintas métricas, pero nosotros utilizaremos la distancia euclídea.

1. Por otra parte, este algoritmo tiene una diferencia fundamental con respecto a los otros dos y es que no incluye la funcionalidad de predecir la clasificación para nuevos datos, por lo que, en la implementación, deberemos incluir también un clasificador, que en este caso será el *KNeighborsClassifier* [9]*,* cuyo funcionamiento es muy similar al de *KMeans* en el sentido de que se basa en la distancia entre *cluster* y datos.**Materiales.**
   1. **Sistemas de calibrado**

Una vez tenemos las muestras de la señal de EMG, necesitamos preprocesarlas para optimizar el funcionamiento del algoritmo de *clustering.*

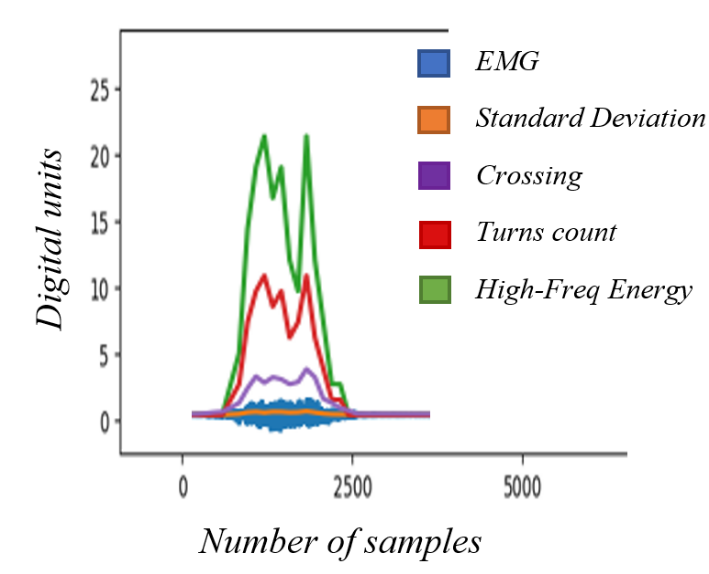
**3.1.1. Extracción de características**

Se conoce como ingeniería de características al hecho de extraer aquellas características de nuestros datos para que nuestros algoritmos funcionen mejor.

Para facilitar la detección de contracciones en nuestras muestras, se han calculado varias características útiles, en lugar de trabajar con la señal de EMG directamente. De alguna forma, todas las características intentan cuantificar la energía del EMG normalizado en pequeñas ventanas de tiempo. En este trabajo, se ha usado una ventana de *hamming* de 0.1 segundos. Dentro de cada  
ventana se han calculado:

1. La desviación estándar de las muestras ("*Standard* *Deviation*" en la figura 1).
2. El número de veces que se cruza cierto nivel de referencia ("*crossing* *rate*" en la figura 1). Como nivel de referencia se empleó 0.2).
3. El número de máximos superiores a cierto nivel de referencia ("*turns* *count*") El nivel de referencia se fijó a 0.2)
4. La energía de altas frecuencias usando donde *diff* representa la diferencia discreta de nuestros datos ("*High*-*Freq* *Energy*" en la Figura 1).

El nivel de referencia para las características de los puntos 2 y 3 se seleccionó tras visualizar varios registros de distintos voluntarios.



**Figura 1:** Ilustración del procesado de señal de EMG, donde se aprecia la línea rosa cuando no hay contracción y las demás representan la señal del EMG simplificada y ampliada en el momento de la contracción.

**3.1.2. Escalamiento**

Una vez tenemos las muestras de la señal simplificada de EMG, necesitamos preprocesarlas para optimizar el funcionamiento del algoritmo de clustering.

El primer paso, que es fundamental antes de iniciar con el preprocesado de los datos, es hacer un escalamiento [10]. Esto es necesario ya que los algoritmos de clustering usan las distancias entre los puntos para definir los clusters. Si las características no tienen la misma magnitud, aquellos con las escalas más grandes dominarán al calcular las distancias.

Hay muchas formas de reescalar datos para todos los lenguajes de programación destinados al análisis de datos. Utilizaremos una que lo que haga sea transformar los datos para ponerlos entre 0 y 1.

3.1.3. Transformación de características

El objetivo de este punto es aplicar una transformación matemática a nuestros datos para intentar que adopten una forma lo más parecida posible a una distribución normal. Este tipo de tratamiento para los datos es muy útil ya que pueden corregir problemas de posibles valores atípicos.

Para este estudio probaremos los siguientes tipos de transformaciones, inspirados por las transformaciones de Box-Cox [11]. Los pequeños valores añadidos a las transformaciones se emplean para evitar divergencias cuando los datos son 0 (aunque en la transformación de raíz no es estrictamente necesario, se hace igualmente por simplificar la implementación):

* Transformación logarítmica:

*Datos\_log =*

* Transformación de raíz:

*=*)

* Transformación recíproca:

*Datos\_reci =*

**3.1.4. Reducción de dimensionalidad**

Los sets de datos que obtenemos del archivo de simplificación de la señal de EMG, están compuestos por todas esas características que extrajimos y que se combinan para formar cada punto del EMG. Muchas veces se da que estas características están correlacionadas, por lo que puede ser importante aplicar una reducción de dimensionalidad para optimizar los cálculos, teniendo en cuenta la cantidad de información útil que se podría perder. Para ello realizaremos un análisis de componentes principales [12] (PCA). Lo que hace este tipo de técnica es precisamente analizar las dimensiones de los datos que no aportan demasiada información importante para reducir el set de datos, optimizando el rendimiento general del programa.

* 1. **Evaluación de los sistemas de calibrado**

Para que el algoritmo de clustering funcione correctamente, es necesario dividir el set de datos en un conjunto de prueba y un conjunto de entrenamiento [13]. En nuestro caso estableceremos que un 70% es para prueba y el 30% restante para entrenamiento.

Es importante mencionar que, en nuestro caso, como queremos crear un calibrador para una persona en concreto, no debemos mezclar los sets de prueba y entrenamiento de diferentes personas para entrenar al algoritmo. Es decir, el algoritmo solo entrenará con los datos de entrenamiento de la persona en cuestión y predecirá con los datos de prueba de esa misma persona.

Para evaluar el rendimiento de los algoritmos haremos uso del conocido *Davies* *Bouldin* *Score*[14]. Esta puntuación se obtiene a partir de medir el promedio de la separación entre cada *cluster* con su *cluster* más similar.

En este caso, tenemos que estudiar la puntuación de todas las combinaciones de todos los modelos de *clustering*, con la posibilidad de hacer o no reducción de dimensionalidad y con todas las opciones de transformación de variable. De esta manera obtendremos una combinación que nos dará un *Davies Boulding Score* más bajo, y por tanto la que determinaremos como la mejor para implementarla en el juego del dinosaurio.

Para hacer esto, hacemos uso de varios bucles anidados para ir comprobando cada posible combinación para cada muestra de EMG que tenemos. Posteriormente pasamos a promediar las puntuaciones obtenidas de cada muestra con respecto a cada combinación para obtener un resultado general sobre el cual poder sacar conclusiones verídicas.

* 1. **Validación**
     1. **Estudio del número de *clusters***

En las secciones 2.2, se asume que los datos se agrupan en dos grupos, por lo que es interesante verificar dicha asunción o si, por el contrario, el número para un agrupamiento óptimo es otro.

Para visualizar esto, podemos utilizar el algoritmo del codo [15], en el que se valora la inercia obtenida por el método *KMeans*. En este tipo de algoritmo se ve representado como el mayor cambio en la inercia se da cuando tenemos el número óptimo de *clusters*. Por esa forma que toma la línea en el cambio brusco, a este algoritmo se le da el nombre de “algoritmo del codo”.

También podemos hacer un estudio del número óptimo de *clusters* basándonos en el algoritmo de agrupamiento jerárquico, es decir teniendo en cuenta la distancia entre los distintos “*subclusters*”. Esto lo hacemos creando un dendrograma [15].

* + 1. **Implementación del sistema en el juego**

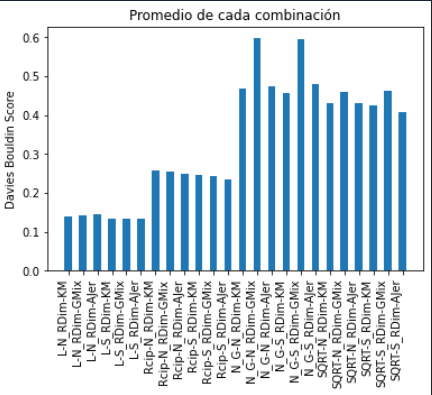
Para verificar que el calibrador funciona correctamente, se implementa en una interfaz paciente-ordenador. En este caso se utilizará para controlar cuándo salta o no el dinosaurio del juego *T-Rex* de *Chrome*, dependiendo de cuándo se detecta una contracción en el paciente. El calibrador está diseñado para que esa detección de contracciones se ajuste automáticamente a cada paciente.

Para llevarlo a cabo, en la interfaz se debe añadir una opción de entrenamiento donde se entrenará al algoritmo a partir de los datos destinados a ello en el *train-test split*, y preprocesados previamente. Posteriormente debemos implementar una función para que el algoritmo prediga si se están produciendo contracciones o no en los nuevos datos que se le están pasando (los datos de las contracciones mientras se juega). El algoritmo deberá detectar si esos nuevos datos pertenecen al clúster de contracción o de no contracción para hacer saltar o no al dinosaurio. Para ello, se deberá detectar cuál es el *cluster* asignado a las contracciones. En la parte de entrenamiento hay que fijarse en las medias del algoritmo, detectando como la media del cluster asignado a la contracción es mayor que la otra. De esta manera podemos utilizar una variable que se utilizará en el momento de la decisión de hacer saltar o no al dinosaurio.

1. **Resultados**

***4.1.1. Resultado del promedio de cada combinación***

La figura 4 muestra el resultado de la comparativa entre las distintas opciones de preprocesado de datos y clustering (sección 3). En el eje vertical se representa el *Davies Boulding Score* (sección 3.2) y en el horizontal las distintas combinaciones de métodos de preprocesado (sección 3.1).



**Figura 4:** Distintas combinaciones de métodos de preprocesado de datos y *clustering*.

***4.1.2. Estudio del número de clusters* por algoritmo del codo**

Gráfico, Gráfico de líneas

Descripción generada automáticamente

**Figura 5:** Resultado gráfico del algoritmo del codo.

La figura 2 muestra el resultado de realizar el algoritmo del codo (sección 3.3.1) sobre nuestros sets de datos. Este tipo de algoritmo muestra el cambio de pendiente más brusco en el número de clusters óptimo. En este caso se ve como el cambio más brusco está en el número 2, lo que significa que efectivamente es 2 el número óptimo de *clusters* para nuestra muestra.

***4.1.3. Estudio del número de clusters* por dendrograma**

Imagen que contiene Histograma

Descripción generada automáticamente

**Figura 6:** Resultado gráfico del estudio del dendrograma*.*

En la figura 3 observamos el producto de realizar un dendrograma (sección 3.3.1) sobre nuestros datos. En este caso también observamos como la distancia más larga se da cuando tratamos de unir los 2 *clusters* principales, por lo que también es 2 el número óptimo.

1. **Discusión**

Como se puede observar en la Figura 4, la puntuación óptima es la obtenida aplicando el logaritmo, la reducción de dimensionalidad y el *Gaussian* *Mixture* *model.* Sin embargo, se puede observar como en realidad parece que el hecho de aplicar el logaritmo y la reducción de dimensionalidad es lo que verdaderamente hace que la puntuación sea buena ya que las diferencias entre las puntuaciones de los distintos métodos de clustering no son significativas.

Esto significa que haciendo más pruebas nos podría salir que es mejor opción usar otro algoritmo de clustering debido a que la diferencia es muy pequeña.

Poniendo a prueba el calibrador con los mismos voluntarios que utilizamos para recoger las muestras (sección 2.1.2) , se puede observar como las contracciones se adaptan a todo tipo de pacientes. Sin embargo, se puede observar un minúsculo *delay* que se asume que depende del procesador del ordenador debido a la gran cantidad de datos.

1. **Conclusión**

En este estudio se ha tratado de crear un calibrador automático basado en *machine learning* con el fin de implementarlo en el controlador de una interfaz paciente-ordenador por señales de EMG, para que se adapte a cualquier paciente automáticamente. Para ello, se ha debido hacer un estudio de cuál es el mejor algoritmo de *clustering* para nuestro caso y las técnicas de preprocesado de datos que mejor se ajustan a los algoritmos. En consecuencia, se obtuvieron muestras de señales de EMG de diferentes voluntarios para comprobar que la opción elegida era la óptima para diferentes muestras y usuarios.

Los resultados de la Figura 4 sugieren que los mejores resultados se obtienen con el preprocesamiento LS-RDIM (hacer el logaritmo y reducción de dimensionalidad), mientras que el algoritmo de clustering no tiene un impacto relevante en el rendimiento.

Los resultados de las figuras 5 y 6 sugieren que la asunción de que los datos se pueden agrupar binariamente (si hay contracción o no) es correcta ya que muestran que el número óptimo de *clusters* es 2.

1. **Referencias**
2. La E, Mar E, Grau A, et al. LAS REDES BAYESIANAS COMO SISTEMAS de SOPORTE a LA DECISIÓN. Accessed December 1, 2021. https://lsi2.ugr.es/~rosana/investigacion/files/abad-efsi02.pdf
3. ‌ De la Hoz Manotas, A. K., Martínez-Palacio, U. J., & Mendoza-Palechor, F. E. (2013). Técnicas de ML en medicina cardiovascular. Memorias, 11(20), 41-46.
4. Velasco, M. A., Clemotte, A., Raya, R., Ceres Ruiz, R., & Rocón, E. (2015). MouseField. Técnica de ayuda al apuntamiento y selección en un interfaz persona-computador basado en el movimiento de cabeza para personas con parálisis cerebral.
5. Guerreiro, J., Martins, R., Silva, H., Lourenço, A., & Fred, A. L. (2013, July). BITalino-A multimodal platform for physiological computing. In ICINCO (1) (pp. 500-506).
6. biosignalsplux | OpenSignals. Biosignalsplux.com. Published 2020. Accedió en septiembre 21, 2021.

https://biosignalsplux.com/products/software/opensignals.html

1. Cambronero, C. G., & Moreno, I. G. (2006). Algoritmos de aprendizaje: knn & kmeans. Intelgencia en Redes de Comunicación, Universidad Carlos III de Madrid, 23. Accedió en septiembre 25, 2021.
2. Instituto de Informatica UACh. [INFO337] Modelos de mezcla de Gaussianas parte 1. YouTube. Published online December 10, 2020. Accedió en octubre 15, 2021. ttps://www.youtube.com/watch?v=s3sTSA\_GXV8&t=5106s&ab\_channel=InstitutodeInformaticaUACh
3. Ligdi González. Algoritmo Agrupamiento Jerárquico - Práctica - Aprende IA. Aprende IA. Published November 10, 2020. Accessed November 21, 2021. https://aprendeia.com/algoritmo-agrupamiento-jerarquico-practica/. Accedió en octubre 1, 2021.
4. sklearn.neighbors.KNeighborsClassifier. scikit-learn. Published 2021. Accedió en noviembre 5, 2021. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
5. sklearn.preprocessing.minmax\_scale. scikit-learn. Published 2021. Accedió en noviembre 9, 2021. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.minmax\_scale.html
6. ‌ Pita Fernández, S., & Pértega Díaz, S. (1997). Relación entre variables cuantitativas. Cad Aten Primaria, 4, 141-4. Accedió en septiembre 19, 2021.
7. Kiko Correoso. Análisis de componentes principales con python – Pybonacci. Pybonacci.org. Published October 8, 2012. Accedió en octubre 16, 2021. https://pybonacci.org/2012/10/08/analisis-de-componentes-principales-con-python/
8. sklearn.model\_selection.train\_test\_split. scikit-learn. Published 2021. Accedió en octubre 19, 2021. https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.train\_test\_split.html
9. sklearn.metrics.davies\_bouldin\_score. scikit-learn. Published 2021. Accessed November 21, 2021. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.davies\_bouldin\_score.html#. Accedió en septiembre 30, 2021
10. ‌‌Villardón, J. L. V. (2007). Introducción al análisis de clúster. Departamento de Estadística, Universidad de Salamanca. 22p. Accedió en septiembre 25, 2021.