

Detección de estímulos en tiempo real

Alberto Altozano Fernández, Antonio Estebanez Yepes
Juan Antonio Martos Navarro y Marcos Martínez Jiménez

Abstract. El análisis de datos funcionales es un campo de interés creciente en Aprendizaje Automático. En este trabajo se aplicaron distintas técnicas de reducción dimensional para identificar los atributos más relevantes en un problema de detección de estímulos en tiempo real. Para ello se enventaron la series temporales de varios sensores y se clasificaron en base a sus atributos originales o a los productos de las técnicas de reducción dimensional. Finalmente, el error cometido se valoró mediante la reconstrucción del estímulo en función del tiempo, obteniendo resultados aceptables incluso al usar un número reducido de los atributos originales.

1 Introducción

En los últimos años la disponibilidad de datos funcionales ha aumentado enormemente, principalmente a través de sensores en tiempo real [1]. Aunque recientemente se han estudiado técnicas que facilitan analizar este tipo de datos, aproximaciones como el enventanado de las series de datos siguen siendo relevantes.

En este trabajo se estudia un dataset que incluye mediciones de varios sensores de gas, así como sensores de temperatura y humedad, expuestos a estímulos de vino, plátano o a ningún estímulo durante parte de su actividad [2]. Uno de los principales objetivos en aplicaciones de aprendizaje automático con sensores es predecir la presencia o ausencia de un estímulo en tiempo real.

Éste es el objetivo del presente trabajo, predecir si en un instante dado de una serie temporal hay un estímulo de vino, de plátano, o ningún estímulo. Además, se busca identificar los sensores y características de los mismos que mejor predicen la presencia de estímulos, lo que podría permitir reducir el coste de un hipotético sistema detector al reducir el número de sensores necesario.

2 Análisis exploratorio de los datos

Antes de diseñar el protocolo experimental de este trabajo se realizó un análisis exploratorio que incluyó: 1) inspección de *missing values*, 2) distribuciones de los atributos, 3) correlaciones entre atributos y 4) detección de *outliers*.

La inspección de missing values reveló que la serie 95 no aparece en el dataset, pero todas las demás series sí aparecen y están completas.

A continuación se analizó la distribución de cada atributo. Todos los atributos presentan distribuciones que se pueden aproximar razonablemente mediante una normal. Esto respalda la aplicación de técnicas de reducción dimensional PCA y LDA que asumen la normalidad de los datos.

Por otro lado, las correlaciones entre los atributos extraídos para las ventanas (ver Sección 3) permitieron identificar conjuntos de atributos altamente

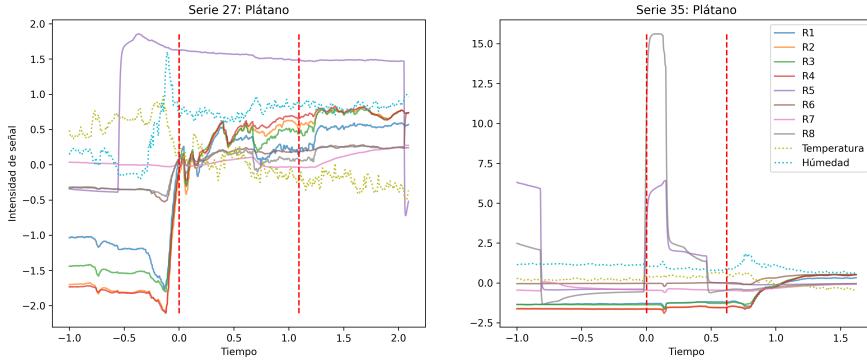


Figura 1: Intensidad de señal para los 8 sensores y la temperatura y humedad en las series 27 (Izda.) y 35 (Dcha.) como ejemplo de las consideradas *outliers*. La zona etiquetada de estímulo corresponde al intervalo rojo.

correlados entre sí (e.g. máximo, media y desviación de R7 y R8), y simplificar heurísticamente las componentes principales de PCA en mPCA.

Finalmente se identificaron como *outliers* las series temporales que presentaban valores extremos en algunos de los sensores. Las series identificadas fueron: 24, 26, 27, 33, 35, 36, 49, 68 y 86 (dos de ellas se muestran en la Figura 1 como ejemplo del tipo de series consideradas *outliers*). Dichas series se eliminaron en los conjuntos sin *outliers*, pero sus resultados se compararon con los de los conjuntos originales, ya que al desconocer los detalles de la toma de datos no se puede garantizar que realmente corresponden a valores erróneos.

3 Metodología

La Figura 2 describe el esquema general del análisis realizado. El dataset original se dividió inicialmente en un conjunto de entrenamiento con 66 series temporales y uno de test con 33. De acuerdo a los datos identificados como outliers en la Sección 2, también se obtuvieron conjuntos de entrenamiento y test sin outliers en base a la misma división de series temporales que en el caso anterior.

Para predecir la presencia o ausencia de estímulos en cada instante de tiempo, las series temporales se dividieron en ventanas solapantes caracterizadas por los siguientes parámetros:

1. Longitud Δt : duración de la ventana.
2. Offset t_o : diferencia entre el tiempo de inicio de una ventana y el de la ventana siguiente.
3. Retardo t_r : diferencia entre el tiempo en que se supone que empieza el estímulo y el momento en que empieza a afectar a los sensores.

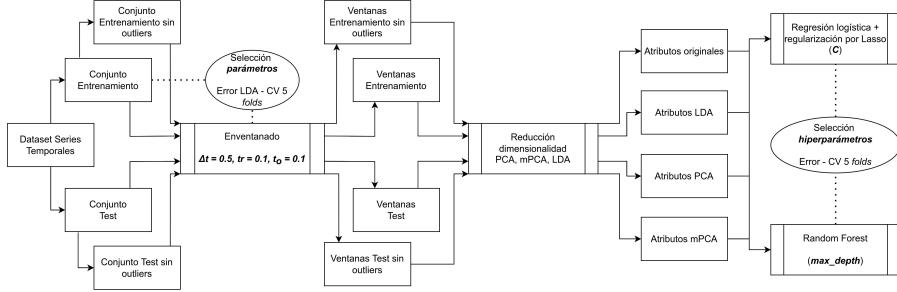


Figura 2: Esquema del análisis realizado. La reducción dimensional se aplicó a cada uno de los dos conjuntos de ventanas (con y sin outliers) y los métodos de clasificación se aplicaron a cada uno de los conjuntos resultantes.

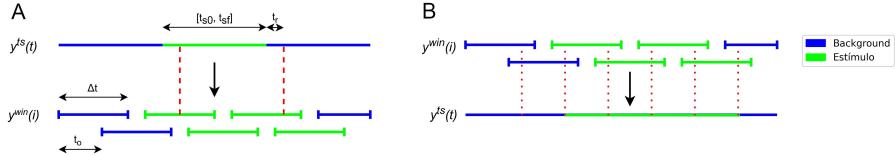


Figura 3: Esquema del proceso de enventanado y etiquetado de las ventanas (A) y de la reconstrucción del estímulo en cada instante de tiempo de acuerdo a una asignación de clases a las ventanas (B).

Siendo t_0 el tiempo de inicio de una ventana, $[t_{s0}, t_{sf}]$ el intervalo de tiempo en que se aplica el estímulo en la serie temporal correspondiente, y y^{ts} la clase de esa serie temporal, la etiqueta asignada a la ventana y_{win} fue y^{ts} si $[t_0, t_0 + \Delta t]$ solapa con $[t_{s0} + t_r, t_{sf} + t_r]$ y Background en caso contrario. La Figura 3A ilustra el proceso de enventanado con un ejemplo.

Por otro lado, las características asignadas a las ventanas fueron los mínimos, máximos, medias y desviaciones estándar de sus series temporales (la curva de temperatura, humedad, y los 8 sensores).

El error de clasificación dada una predicción de clases de las ventanas se valoró mediante una reconstrucción de la clase correspondiente a cada instante de tiempo $\hat{y}^{ts}(t)$, de forma que tuviera en cuenta el error que produce el enventanado. Dicha reconstrucción consiste en asignar la clase de la ventana al intervalo $[t_0 + \frac{\Delta t}{2} - \frac{t_0}{2}, t_0 + \frac{\Delta t}{2} + \frac{t_0}{2}]$ y completar las zonas que quedan sin clase al principio y al final con la clase de la primera y última ventana respectivamente. La Figura 3B ilustra el proceso de reconstrucción usando como clasificación la asignación de etiquetas mostrada en la 3A. Cabe destacar que aún con el etiquetado original de las ventanas, que correspondería a una clasificación perfecta, se obtiene cierto error por la pérdida de información durante el enventanado.

Los parámetros de enventanado se seleccionaron mediante el error de reconstrucción de un modelo LDA, medido por validación cruzada con 5 *folds* en el conjunto de entrenamiento. De esta forma no solo se tuvo en cuenta la sep-

arabilidad de las clases con cada conjunto de parámetros, sino la pérdida de resolución que se produce con ventanas demasiado grandes. Además, el error utilizado fue la media del error en Background y el error en Estímulo, que aporta el mismo peso a ambas partes independientemente de su duración relativa. De lo contrario ventanas demasiado cortas dan lugar a un gran desequilibrio a favor de Background y permiten reducir el error a costa de la predicción del estímulo.

Los parámetros de enventanado óptimos se usaron para obtener ventanas de los conjuntos con y sin outliers. Cada uno de estos dos conjunto de ventanas se estandarizó y se usó para obtener versiones de menor dimensionalidad por varias técnicas

1. PCA: entrenado sobre el conjunto de datos completo (entrenamiento y test), ya que es un método no supervisado.
2. mPCA: selección manual de atributos en base a la interpretación de los coeficientes de PCA, de modo que representaran las PCs de forma *sparse*.
3. LDA

Finalmente se aplicaron Regresión Logística y Random Forest sobre los conjuntos con y sin outliers en sus versiones originales y con cada reducción de dimensionalidad. Para ello se seleccionaron sus hiperparámetros (coeficiente de regularización, y numero de estimadores y profundidad máxima respectivamente) mediante validación cruzada en el conjunto de entrenamiento.

4 Resultados y Discusión

4.1 Estudio de los parámetros de enventanado

La Figura 4 muestra la curva de error CV para distintos valores de Δt y la separación de clases en LDA resultante para el valor óptimo, $\Delta t = 0.5$. Se puede apreciar que esta medida de error penaliza ventanas demasiado grandes, que separan mejor los datos pero producen error al reducir la resolución temporal, y ventanas demasiado pequeñas, que incrementan el error en Estímulo a cambio de reducir el error de Background. La separación de clases obtenida no es perfecta (destaca el solape de Background y Platano), pero se identifican claramente regiones Enriquecidas en cada clase.

La Figura 5 resume el resultado de aplicar PCA a las ventanas. Se seleccionaron las 6 primeras PCs, necesarias para explicar el 80% de la varianza, y se inspeccionaron sus coeficientes para proponer versiones simplificadas, con uno o dos atributos, que mantuvieran la mayoría de su información de forma *sparse* (mPCs). La separación de clases resultante en las dos primeras componentes es menor en PCA y mPCA que en LDA. También cabe destacar que aunque mPCA separa peor que PCA, los resultados son notablemente similares.

Una vez conocido el valor óptimo de los parámetros de enventanado ($\Delta t = 0.5$ y $t_o = t_r = 0.1$), se guardaron los datasets con las ventanas correspondientes con el fin de usarlos en los métodos de clasificación.

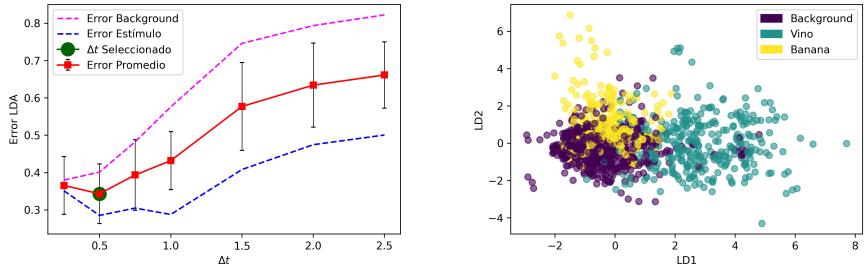


Figura 4: Error de LDA medido por validación cruzada en las ventanas de entrenamiento generadas con $t_o = 0.1$, $t_r = 0.1$ y distintos valores de Δt (Izda.), y separación de las clases en el espacio LDA para el valor óptimo $\Delta t = 0.5$ (Dcha.).

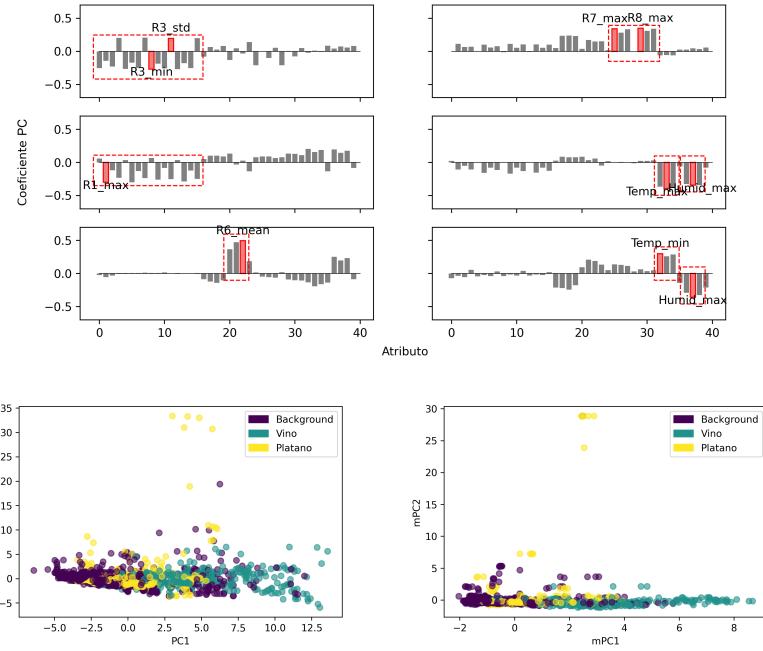


Figura 5: Coeficientes para los distintos atributos de las PCs obtenidas con ventanas de entrenamiento y test (Arriba) y separación de las clases en el espacio PCA (Abajo Izda.) y mPCA (Abajo Dcha.). Las coeficientes que se consideraron representativas de cada PC se marcan en rojo, así como el conjunto de atributos que representan, al estar fuertemente correlacionados con ellos.

4.2 Regresión logística y Random Forest

4.2.1 Selección de hiperparámetros

Se utilizaron los módulos de `sklearn` [3]: `linear_model.LogisticRegression` con una regularización del tipo *elasticnet*, y `ensemble.RandomForestClassifier`, con el criterio de Gini como criterio de división. En el primero, *elasticnet* consiste en una mezcla de regularización l_1 y l_2 (lasso y ridge respectivamente), donde hay que seleccionar los hiperparámetros l_1_ratio (ratio entre penalización lasso o ridge) y C (inversa de la fuerza de la regularización). En el segundo hay que seleccionar el número de árboles y la profundidad (`max_depth`).

El resultado de aplicar validación cruzada sobre todos los datasets para distintos valores de l_1_ratio en la regresión logística es muy similar. Por este motivo se fijó $l_1_ratio = 1$ (lasso), ya que favorece la *sparsity* del resultado. Por otro lado, el número de árboles utilizados para random forest es de 100, ya que no sobreajusta con el número de árboles.

Las Figuras 6 y 7 muestran los errores de validación para distintos valores de C en regresión logística y `max_depth` en random forest respectivamente.

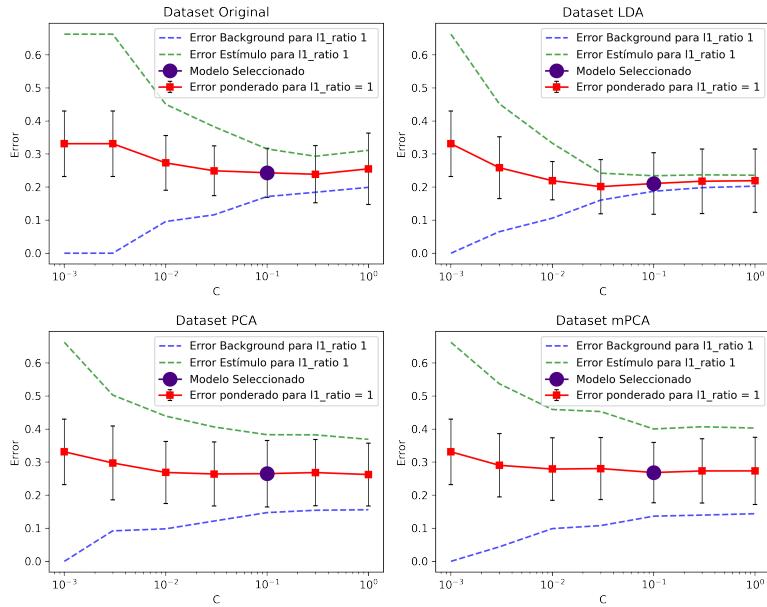


Figura 6: Errores de reconstrucción (Background, Estímulo y Ponderado) del modelo medidos por validación cruzada para distintos valores de C en regresión logística. El modelo seleccionado se marca en morado.

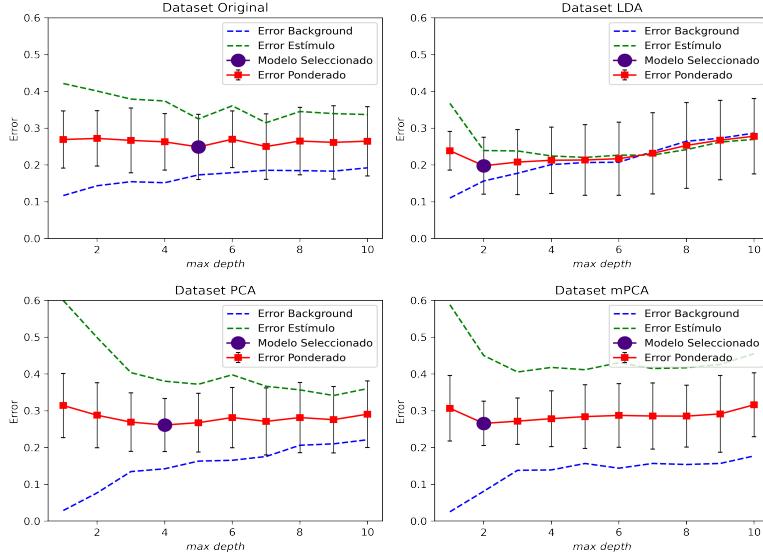


Figura 7: Errores de reconstrucción (Background, Estímulo y Ponderado) del modelo medidos por validación cruzada para distintos valores de profundidad en random forest. El modelo seleccionado se marca en morado.

4.2.2 Clasificación: Entrenamiento y test

Los modelos considerados se entrenaron con los hiperparámetros seleccionados en la Sección 4.2.1. En las figuras 8A y 8B se muestran las matrices de confusión para cada modelo y cada dataset, y en la Tabla 1 sus respectivos rendimientos.

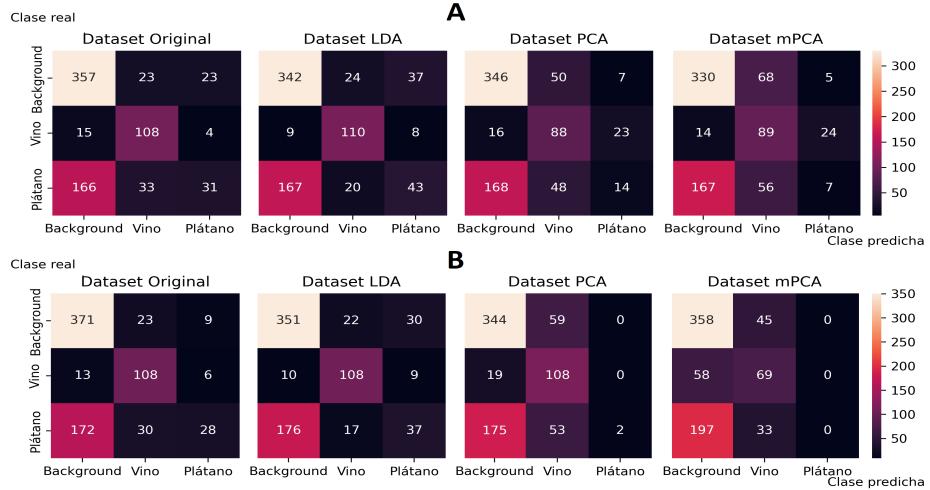


Figura 8: Matrices de confusión de regresión logística (A) y random forest (B).

Tabla 1: Rendimiento cada modelo para cada dataset.

Dataset	Rendimiento Background(%)	Rendimiento Estímulo(%)	Rendimiento Ponderado(%)
Modelo Regresión Logística			
Original	78.19	64.32	71.26
PCA	75.74	57.53	66.63
mPCA	72.37	53.88	63.13
LDA	74.38	68.87	71.63
Modelo Random Forest			
Original	81.17	63.55	72.36
PCA	79.39	54.05	66.72
mPCA	77.20	56.05	66.63
LDA	79.03	66.04	72.53
Modelo nulo: Predicción siempre background			
	100	24.86	62.43

Además, las Figuras 9A y 9B muestran las clases predichas y las reales para el test y entrenamiento de cada uno de los modelos en el espacio LDA.

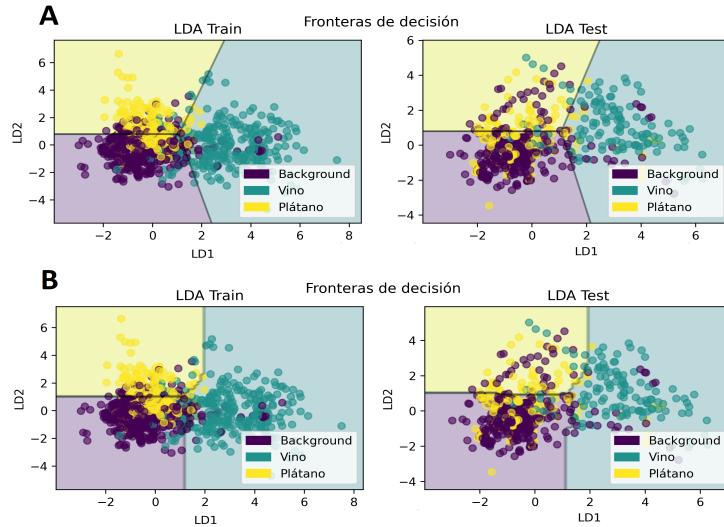


Figura 9: Fronteras de decisión de regresión logística (A) y random forest (B) sobre los datos de entrenamiento (Izda.) y de validación (Dcha.) de LDA.

Los modelos resultantes tienen rendimientos aceptables en la detección del impulso. El mayor error de confusión se da entre Background y Plátano, cuyas ventanas son notablemente difíciles de separar. Finalmente, el dataset mPCA tiene un rendimiento menor que alternativas como LDA, pero implica solo 4 sensores, por lo que podría ser deseable al reducir el coste económico del dispositivo.

References

- [1] J.R. Berrendero, A. Justel, and M. Svarc. Principal components for multivariate functional data. *Computational Statistics Data Analysis*, 55(9):2619–2634, 2011.
- [2] Ramon Huerta, Thiago Mosqueiro, Jordi Fonollosa, Nikolai F Rulkov, and Irene Rodriguez-Lujan. Online decorrelation of humidity and temperature in chemical sensors for continuous monitoring. *Chemometrics and Intelligent Laboratory Systems*, 157:169–176, 2016.
- [3] 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*, 12:2825–2830, 2011.