Uso de técnicas de Saliency para Selección de Características

Brais Cancela
LIDIA Group
Universidade da Coruña
A Coruña, Spain
brais.cancela@udc.es

Verónica Bolón-Canedo
LIDIA Group
Universidade da Coruña
A Coruña, Spain
veronica.bolon@udc.es

Amparo Alonso-Betanzos
LIDIA Group
Universidade da Coruña
A Coruña, Spain
ciamparo@udc.es

João Gama
LIAAD Group
University of Porto
Porto, Portugal
jgama@fep.up.pt

***Resumen*—Las técnicas clásicas de selección de características buscan la eliminación de aquellas características que son irrelevantes o reduntantes, obteniendo un subconjunto de características relevantes, que ayudan a una mejor extracción de conocimiento del problema a tratar, permitiendo modelos de aprendizaje más sencillos y fáciles de interpretar. La gran mayoría de estas técnicas trabajan sobre el conjunto completo de datos, pero no nos proporcionan una información detallada caso por caso, que es el escenario del que partimos en esta propuesta. En este trabajo mostraremos un nuevo método de selección de características, basado en las técnicas de saliency en deep learning, que es capaz de proporcionar un conjunto de características relevantes muestra a muestra, y finalmente proporcionar un subconjunto final de las mismas.**

***Index Terms*—selección de características, deep learning, saliency**

I. Introducción

Con el auge del denominado *Big Data*, el uso de técnicas que permitan reducir la dimensionalidad del espacio de entrada se hace cada vez más necesario. Las técnicas que acometen esta tarea se dividen, habitualmente, en dos grandes grupos: la *selección de características* (feature selection) y la *extracción de características* (feature extraction). En la Figura 1 podemos ver una representación gráfica sobre su funcionamiento.

Por una parte, las técnicas de extracción de características reducen el número de características mediante la combinación de las variables originales. Un ejemplo en *Deep Learning* serían las conocidas como *deep features*, que son las características que se utilizan como entrada de la última capa de una red. De esta manera, estas técnicas son capaces de crear un nuevo conjunto de características, que suele ser más compacto y con una capacidad mayor de discriminación. Esta es la técnica más utilizada en análisis de imágenes, procesamiento de señales o en recuperación de la información.

Por otra parte, las técnicas de selección de características consiguen la reducción de la dimensionalidad mediante la eliminación de características tanto irrelevantes como redundantes. Debido al hecho de que la selección de características mantiene los atributos *originales*, es una técnica especialmente útil para aplicaciones donde los atributos originales son importantes para la interpretación del modelo y la extracción de conocimiento.

Brais Cancela agradece el apoyo de la Xunta de Galicia bajo su programa postdoctoral. También agradecemos su ayuda a NVIDIA, que nos ha facilitado una tarjeta Titan Xp bajo el 'GPU Grant Program'.

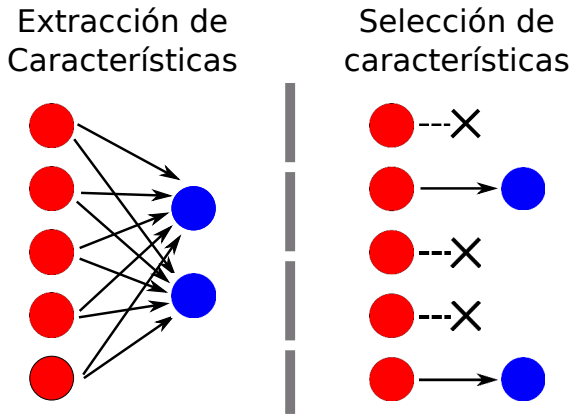

Figura 1. La extracción de características crea nuevas características mediante la combinación del conjunto inicial, mientras que la selección de características elimina aquellas que considera irrelevantes o redundantes, manteniendo inalteradas las restantes.

Uno de los problemas de los que adolece la selección de características es que, a pesar de ayudar a extraer un conocimiento más transparente y explicable acerca de las variables que son relevantes para el problema que estemos tratando, lo hace de una manera global. Esto es, no provee información caso por caso. Veamos un ejemplo: supongamos que tenemos un conjunto de datos de pacientes, y nos interesa predecir si un paciente concreto es propenso a padecer cáncer o no. Los métodos de selección de características son capaces de indicarnos cuáles son las características que más influyen a la hora de realizar dicha clasificación. Pero supongamos que tenemos un paciente, y queremos saber, exactamente, cuáles son las características que le han indicado al sistema que dicho paciente es propenso a padecer cáncer. Los sistemas clásicos de selección de características son incapaces de proveer dicha información.

En este trabajo proponemos abordar el problema de selección de características partiendo de este escenario . Nuestra idea propone la utilización de técnicas que nos indiquen, caso por caso, cuáles son las características de cada uno con la información más discriminante. Una vez detectadas, creare-

mos un algoritmo de selección de características, incluyendo aquéllas con un valor medio de discriminación más alto.

El resto del artículo estará estructurado de la siguiente manera: la sección II explicará el método que vamos a utilizar para la evaluación de la importancia de cada característica; la sección III propondrá nuestro algoritmo de selección de características, basado en redes neuronales; finalmente, la sección IV ofrecerá resultados experimentales sobre un conjundo de datasets públicos, y la sección V propondrá nuestras conclusiones y trabajos futuros.

II. Selección de características en Deep Learning

La literatura científica en el campo de métodos de selección de características que hagan uso de modelos de redes neuronales masivas, tales como por ejemplo, las Redes Convolucionales (CNN), es muy escasa en la actualidad. Uno de los trabajos más interesantes es una variante del LASSO para ser utilizada en CNNs, llamada DeepLASSO [1]. El método introduce un factor multiplicativo a los valores de entrada (llamado *máscara*), sobre el que se aplica un factor de regularización equivalente al del método LASSO (norma 1). Los autores de [1] proponen diferentes regularizaciones (LASSO, elastic Net, etc.) sobre la misma estructura. Las restantes aproximaciones a la selección de características en Deep Learning se englobarían dentro del subconjunto de técnicas denominado como *saliency*.

II-A. Saliency

Saliency es una técnica que surge en la visión por computador, con el objetivo de evaluar la *calidad* de cada uno de los píxeles que componen una imagen. Clásicamente, las redes neuronales se han visto como una caja negra, en la que se obtiene cierta salida a partir de los datos de entrada, pero sin ningún tipo de explicación más o menos transparente sobre cómo se ha llegado a esa solución. Actualmente existen dos maneras de calcular esta información de salida: de una manera semi-supervisada, o totalmente supervisada.

Las primeras técnicas que se utilizaron eran del tipo semi-supervisado [2], [3]. Constan de una red que es entrenada como un clasificador (binario o multiclase). Una vez entrenada la red, y dada una imagen de entrada, se realiza una retropropagación desde la salida hasta la entrada, para averiguar cuáles han sido los píxeles que más han influído en la salida esperada.

Los modelos totalmente supervisados son más recientes [4], [5], y son entrenados de una manera distinta. En este caso, se hace uso de la llamada *segmentación semántica*, ya que la salida de la red tiene el mismo tamaño que la imagen de entrada. Además, para cada imagen sabemos cuáles son los píxeles importantes (por ejemplo, si estamos detectando cierto tipo de objeto, los píxeles importantes son aquellos que pertenecen al obejto en cuestión, mientras que el fondo es considerado irrelevante). El modelo entrena la red para que la salida concuerde con nuestra segmentación previa de píxeles importantes. Estos modelos también suelen ser conocidos como *modelos de atención* [6], [7] cuando se utiliza una Red Neuronal Recursiva (RNN) como último paso para evaluar la calidad de cada píxel.

Las técnicas totalmente supervisadas obtienen mejores resultados, pero tienen un hándicap importante: para entrenar el modelo se necesita saber, a priori, cuáles son las características relevantes de cada uno de los ejemplos de entrada. En el caso de imágenes suele ser sencillo, pero no siempre es posible obtener dicha información, puesto que no siempre se sabe cuáles son las características que contienen la información más discriminante. Por tanto, para nuestro modelo de selección de características, proponemos hacer uso de una técnica de saliency semi-supervisada.

II-B. Aproximación propuesta

Para nuestro modelo vamos a utilizar una generalización de la idea propuesta en [2]. Sea un conjunto de datos de entrada $\mathbf{X} \in \mathcal{R}^{N\times R}$, con N número de muestras y R número de características; y $\tilde{\mathbf{Y}} = f(\mathbf{X};\mathbf{\Theta}) \in \mathcal{R}^{N\times C}$ nuestro clasificador, que puede ser tanto un clasificador linear como una CNN de muchas capas. En este caso C es el número de clases a evaluar, y $\mathbf{\Theta}$ son los pesos del clasificador que necesitan ser ajustados en la etapa de entrenamiento. Por simplificación, asumimos que los valores de $f(\mathbf{X};\mathbf{\Theta})$ son el resultado de aplicar la función softmax, devolviendo la probabilidad de pertenencia a cada clase. Esto es,

$$\sum_{c=1}^{C} \tilde{y}_c^{(i)} = 1 \tag{1}$$

Para entrenar la red se minimizará una función de pérdida $\ell(\mathbf{\Theta}; f, \mathbf{X}, \mathbf{Y})$, donde $\mathbf{Y} \in \mathcal{R}^{N\times C}$ es la codificación binaria de la clase esperada (*one-hot encoding*). En nuestro caso, minimizaremos la función de entropía cruzada (*cross-entropy*), definida como

$$\ell(\mathbf{\Theta}; f, \mathbf{X}, \mathbf{Y}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_c^{(i)} \log\left(f(\mathbf{x}^{(i)};\mathbf{\Theta})_c\right) \tag{2}$$

En el caso de nuestra aproximación, se puede utilizar en conjunto con la técnica DeepLASSO [1], en cuyo caso la función de coste se modificaría de la siguiente manera

$$\ell(\mathbf{\Theta}, \psi; f, \mathbf{X}, \mathbf{Y}) = \|\psi\|_1 - \frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_c^{(i)} \log\left(f(\psi * \mathbf{x}^{(i)};\mathbf{\Theta})_c\right), \tag{3}$$

donde ψ es el vector *máscara*.

La idea de saliency es similar a la que se utiliza para actualizar los pesos $\mathbf{\Theta}$, calculando su derivada con respecto a la función de coste, esto es

$$\mathbf{\Theta} \leftarrow \mathbf{\Theta} - \alpha\frac{\partial \ell}{\partial \Theta}, \tag{4}$$

siendo α el factor de aprendizaje.

En cuanto a su significado numérico, los valores altos en magnitud de la derivada indican que dicho peso está influyendo negativamente en el funcionamiento correcto del sistema, mientras que valores cercanos a 0 indican que el peso está bien ajustado. En el caso del saliency, nos interesaría que las

características con magnitudes altas sean las que contienen información más significativa, mientras que las cercanas a 0 sean características irrelevantes. Nuestra función de saliency se define como

$$\sigma(\mathbf{x}^{(i)}; \mathbf{\Theta}, f, y^{(i)}) = \left| \frac{\partial g(\mathbf{x}^{(i)}; \mathbf{\Theta}, f, y^{(i)})}{\partial \mathbf{x}^{(i)}} \right|, \quad (5)$$

esto es, nos interesa saber el gradiente de los ejemplos de entrada con respecto a una función, que llamaremos *función de ganancia* (g), que devuelve su valor máximo cuando la clasificación es perfecta ($\ell = 0$), y valores cercanos a 0 cuando el clasificador falla. En el caso de la función de coste de entropía cruzada que estamos utilizando (Eq. 2), definimos la función de ganancia como

$$g(\mathbf{x}^{(i)}; \mathbf{\Theta}, f, y^{(i)}) = -\beta y^{(i)} \log \Big(1 - f(\mathbf{x}^{(i)}; \mathbf{\Theta})\Big), \quad (6)$$

donde β es un hiperparámetro que controla el decaimiento de la función de ganancia. Por defecto, $\beta = 1$. Para evitar un gradiente infinito, podamos aquellos valores de $f(\mathbf{x}^{(i)}; \mathbf{\Theta}) = 1$. Por defecto,

$$f(\mathbf{x}^{(i)}; \mathbf{\Theta}) = \text{mín}\{1 - e^{-8}, \quad f(\mathbf{x}^{(i)}; \mathbf{\Theta})\} \quad (7)$$

Es importante remarcar que esta técnica es similar a la propuesta en [2]. Se diferencia en un único punto: mientras que el método en [2] es específico para un problema de clasificación, dado que descomponen la última capa de la red para aplicar su idea, nuestro modelo opera directamente como una función matemática de ganancia, lo que lo hace susceptible de poder ser utilizado en otro tipo de problemáticas, como los modelos de regresión. Otra ventaja de nuestra aproximación es que, dado que se aplica sobre una función de ganancia del clasificador, el gradiente es cercano a 0 cuando el clasificador se equivoca. Por tanto, el sistema nos dice que no hay características relevantes, haciéndolo robusto ante ejemplos que no son clasificados correctamente por la red.

III. SELECCIÓN DE CARACTERÍSTICAS USANDO SALIENCY

Nuestro método de selección de características tendrá un comportamiento de tipo *ranker*, es decir, devolverá un vector ordenado de todas las características en función de su importancia. El esquema de nuestra aproximación puede verse en el Algoritmo 1.

El algoritmo propuesto tiene dos hiper-parámetros: $\epsilon \geq 0$, como control de parada, y $1 \geq \gamma > 0$, que controla el porcentaje de características *vivas* que van a ser mantenidas para la siguiente iteración.

Comenzamos entrenando la red neuronal f con todos los datos de entrenamiento. Luego, por cada clase calculamos el saliency de cada uno de los ejemplos, para luego normalizar mediante la norma 1. De esta manera para cada clase tenemos la probabilidad de que cada una de las características sea relevante para la clasificación. Una vez obtenidas las probabilidades de todas las clases, se suman y se ordenan, obteniendo el ranking de importancia de las características.

Datos: $\mathbf{X}, \mathbf{Y}, f, \ell, \mathbf{\Theta}, \gamma, \epsilon$, *reps*
Resultado: ranking de características $\mathbf{r}$
$n_f \leftarrow R$;
$\mathbf{r} \leftarrow [1 \dots n_f]$;
mientras $n_f > \epsilon > 1$ **hacer**
  $\hat{\mathbf{X}} \leftarrow \mathbf{X}$;
  $\hat{\mathbf{X}}[:, \mathbf{r}[n_f + 1 : R]] \leftarrow 0$;
  $\sigma_{fs} \leftarrow \text{zeros}(n_f)$;
  para $r \leftarrow 1$ **a** *reps* **hacer**
    Inicializar $f(\hat{\mathbf{X}}; \mathbf{\Theta})$;
    Entrenar $f(\hat{\mathbf{X}}; \mathbf{\Theta})$ dado $\mathbf{Y}$;
    para $c \leftarrow 1$ **a** C **hacer**
      $\sigma_c \leftarrow \sum_{i_c=1}^{N_c} \sigma(\mathbf{x}^{(i_c)}; \ell, \mathbf{\Theta}, f, y^{(i_c)})$;
      $\sigma_{fs} \leftarrow \sigma_{fs} + \frac{\sigma_c}{\|\sigma_c\|_1}$;
    fin
  fin
  index $\leftarrow$ argsort(σ_{fs}, descend);
  $\mathbf{r}[1 : n_f] \leftarrow \mathbf{r}$[index];
  $n_f \leftarrow int(n_f * \gamma)$;
fin

Algoritmo 1: Selección de características usando saliency

No obstante, hay que tener en cuenta que las redes neuronales son propensas al *sobre-entrenamiento* (overfitting). Esto haría que, en caso de que se eliminaran ciertas características, el reentrenamiento de la red cambiaría sustancialmente los valores de los pesos de la misma. Por esta razón, utlizaremos dos variables: (a) *reps*, que entrenará la red un número determinado de veces, para evitar valores atípicos; y (b), una vez obtenido el ranking de características, utilizamos el hiper-parámetro γ para eliminar un porcentaje de las características, con objeto de volver a reentrenar la red y ajustar el orden de las características áun activas. El algoritmo se detendrá cuando el número de características activas sea inferior al hiper-parámetro ϵ.

Por lo tanto, la complejidad del algoritmo es variable, pues depende completamente del parámetro γ. En los casos extremos, tenemos que si $\gamma = 0$, entonces la complejidad es lineal ($\mathcal{O}(Nreps)$), dependiendo sólo del número de ejemplos y de repeticiones. Sin embargo, con un $\gamma \approx 1$, la complejidad pasa a depender del número de variables ($\mathcal{O}(RNreps)$), pues necesitamos entrenar tantas redes como características distintas tengamos.

IV. RESULTADOS EXPERIMENTALES

Para evaluar el comportamiento de nuestra aproximación, hemos utilizado los 5 datasets propuestos en el *NIPS 2003 Features Selection challenge* [1]. Se trata de una serie de datasets sintéticos, diseñados con el objetivo de medir la calidad de los resultados obtenidos por los métodos de selección de características. Las características específicas de cada uno de estos datasets se muestran en la tabla I. Es un conjunto

[1] http://clopinet.com/isabelle/Projects/NIPS2003/

Cuadro I
DATASETS UTILIZADOS.

Conjunto	# ejemplos (entr., test)	# total características	# características válidas	% características válidas	ratio positivos/negativos
Arcene	(88, 112)	10000	7000	0.7	1.0
Dexter	(300, 300)	20000	9947	0.5	1.0
Dorothea	(800, 350)	100000	50000	0.5	0.11
Gisette	(6000, 1000)	5000	2500	0.5	1.0
Madelon	(2000, 600)	500	20	0.04	1.0

de datasets particularmente complejo porque incluye diversos tipos de casuísticas: pocos ejemplos de entrenamiento (Arcene), datos desbalanceados (Dorothea) o pocas características válidas (Madelon).

El algoritmo propuesto tiene la ventaja de que es capaz de calcular el ranking de las características al mismo tiempo que entrena el clasificador. Por tanto, se podría pensar que el subconjunto de características seleccionado es dependiente de la arquitectura. Por este motivo, hemos decidido separar la obtención de características del entrenamiento de la red, usando dos arquitecturas distintas para cada cometido. Con este punto creemos poder realizar una comparación más justa con otros métodos que no poseen esta característica, como son los basados en el análisis de información mutua [8].

IV-0a. Arquitectura para la obtención del ranking de características: Para este cometido hemos decidido utilizar una red neuronal totalmente conectada con 3 capas ocultas de 150, 100 y 50 elementos, respectivamente. Utilizamos *Batch Normalization* (BN) [9] seguido de la activación $ReLU(x) = max(0, x)$ en cada capa; como salida usamos un softmax, y utilizamos la entropía cruzada (Eq. 2) como función de coste.

Asimismo, usamos un *weight decay* de 0,001 para todos los pesos, con el objeto de eliminar el sobreentrenamiento. Entrenamos dicha red para un total de 100 iteraciones sobre el conjunto de entrenamiento, usando Adam [10] como optimizador. En el caso de tener datos desbalanceados, aumentamos el número de muestras hasta igualar los ejemplos disponibles para cada clase.

Para la creación y entrenamiento de esta red se ha utilizado el framework Keras [2] sobre Tensorflow [11].

IV-0b. Arquitectura del clasificador: Dado el pequeño número de muestras que tenemos en los conjuntos, nos hemos decantado por utilizar como clasificador una *Máquina de vector soporte* (SVM) con kernel gaussiano (RBF). Hemos usado el hiper-parámetro $C = 1$, sobre la implementación existente en la librería *scikit-learn* de Python.

IV-A. Efecto del parámetro γ

En primer lugar queremos conocer cuál es el efecto que producen los hiper-parámetros γ y *reps* (ver sección III y algoritmo 1) . En el caso del parámetro γ, hemos ejecutado nuestro algoritmo utilizando para ello sólo una repetición para cada conjunto de características (*reps* $= 1$). En la Fig. 2 podemos observar cómo la precisión del algoritmo mejora conforme vamos aumentando el valor de γ. Esto es debido a que, dado que el número de ejemplos de cada dataset es reducido, se produce mucho sobre-entrenamiento en la red. Dicho sobre-entrenamiento causa que determinadas características pasen a ser relevantes, únicamente debido a la inicialización del modelo.

Como caso inusual, en el dataset *Dorothea* se produce el efecto contrario. Esto puede ser debido tanto a la naturaleza de las características (binarias) del dataset, como a lo desbalanceado de los datos (muchos más ejemplos negativos que positivos).

IV-B. Efecto del parámetro reps

Como comentamos anteriormente, el sobre-entrenamiento introduce una variabilidad indeseada en la valoración de las características. Así como el uso del hiper-parámetro γ es una manera de tratar de solventar este inconveniente, éste también podría ser mitigado aumentando el número de veces que entrenamos la red, esto es, aumentando el parámetro *reps*.

Por tanto, hemos ejecutado nuestro algoritmo, fijando para ello el parámetro $\gamma = 0$, y variando el número de repeticiones. La Fig. 3 muestra los resultados obtenidos. Contrariamente a lo ocurrido con el hiper-parámetro γ, el aumento del número de repeticiones no conlleva una mejora sustancial de los resultados obtenidos. De igual manera, en el dataset *Dorothea* podemos ver que se produce el mismo efecto que ya habíamos presenciado con el hiper-parámetro γ.

IV-C. Evaluación de rendimiento

Para probar el rendimiento de nuestra propuesta, hemos decidido comparar nuestro método con una serie de algoritmos clásicos de selección de características: *MIM* [12] y *ReliefF* [13], en sus implementaciones existentes en la librería Weka [14]. Pese a que son algoritmos ya clásicos, siguen siendo utilizados en la actualidad por su probada eficacia. Los hemos escogido por ser métodos que devuelven un ranking de características, de la misma manera que lo hace nuestra propuesta. Además de estos algoritmos en Weka, hemos preparado también una variación del DeepLASSO [1] para poder usarlo en el estudio comparativo. El funcionamiento de éste es equivalente al de nuestro algoritmo, con la salvedad de que en lugar de una función de saliency, utilizaremos el valor absoluto de los valores de la máscara γ descrita en Eq. 3 como la medida de calidad de las características.

Como hemos mencionado anteriormente, hemos decidido separar la obtención del subconjunto de características relevantes del entrenamiento del clasificador, usando dos arquitecturas distintas para cada cometido. Con este punto creemos poder

[2]https://keras.io/

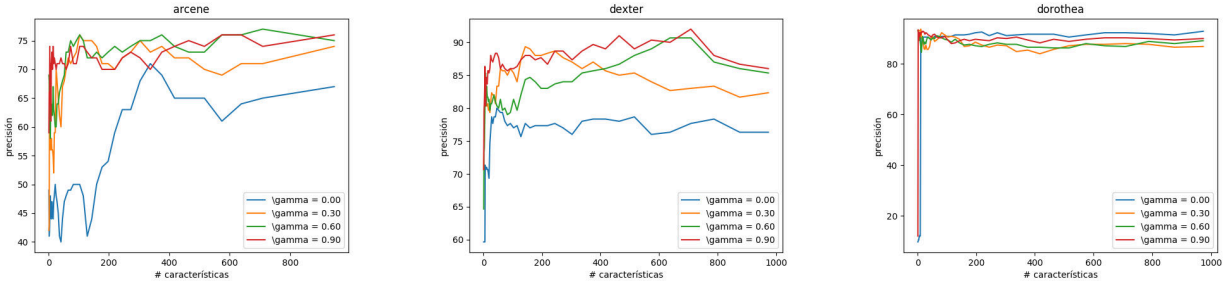

Figura 2. Efecto del hiper-parámetro γ sobre el algoritmo. Conforme su valor se acerca a 1, la selección de características se vuelve más precisa, especialmente cuando el número de características utilizadas es reducido.

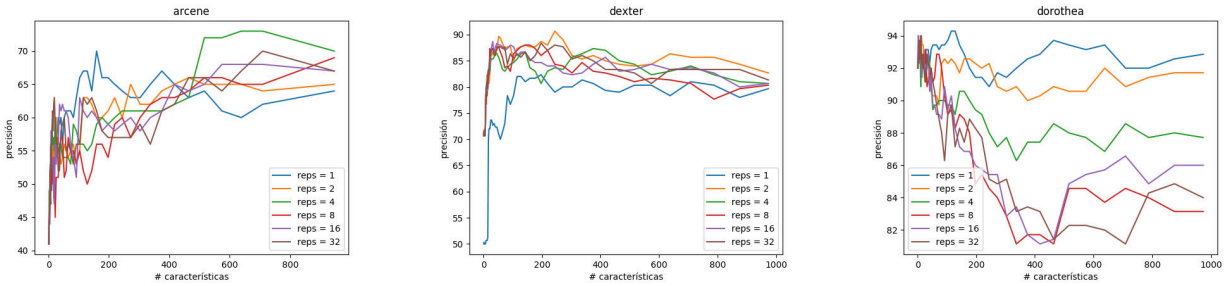

Figura 3. Efecto del hiper-parámetro *reps* sobre el algoritmo. Al contrario que con γ, su aumento no parece suponer una mejora significativa en los resultados.

realizar una comparación más justa contra los métodos *MIN* y *ReliefF*.

En la tabla II podemos observar los resultados obtenidos. En general, nuestra aproximación funciona de una manera análoga a los algoritmos basados en información mutua, algo remarcable dado que estamos utilizando dos arquitecturas distintas para el cálculo del ranking y la clasificación. Podemos observar como la técnica de DeepLASSO no obtiene buenos resultados cuando no utilizamos la misma arquitectura tanto para ranking como para la clasificación. Asimismo, también podemos observar como, en aquellos datasets que contienen más ejemplos (Gisette y Madelon), nuestro método es el aue ofrece los mejores resultados. Asimismo, el valor del parámetro *Área bajo la curva* (AUC) de nuestra propuesta es mayor es prácticamente todos los conjuntos, lo que sugiere una robustez mayor en la selección de características.

Analizando un ranking de las posiciones medias en las que se situarían los cuatro métodos comparados, veríamos que nuestra propuesta es el que tiene mejor media en cuanto a precisión, con un puesto 1,6 de media entre los cuatro métodos, y es además es el segundo de de los métodos, a muy poca diferencia de DeepLasso, que consigue mejores resultados con un subconjunto de características seleccionado más bajo (véase la tabla III). Nótese que la diferencia es bastante importante en conjuntos como Dexter, en los que consigue la segunda mejor precisión (con una diferencia de 0,04, pero con 28 características en vez de las 103 del mejor método en precisión, que en este caso es MIM). Así pues nuestra propuesta es bastante estable, consguiendo los mejores resultados de precisión en media, con un número de características bajo.

V. Conclusiones y Trabajo Futuro

En este paper hemos propuesto un nuevo algoritmo de selección de características basado en *saliency*. Dado un clasificador, y la creación de una función de ganancia, es posible conocer, para cada ejemplo, cuáles son las características más importantes a la hora de clasificar su comportamiento. Esta propiedad, por sí sola, añade una capacidad a nuestro algoritmo de la que carecen los métodos clásicos de selección de características. Asimismo, se trata de un método muy flexible, puesto que permite su uso en casi cualquier clasificador usado en la actualidad (SVM, CNN, ...). Por tanto, se trata de una herramienta muy útil para la selección de características en entornos de Big Data.

Como trabajo futuro, planteamos dos alternativas diferentes. En primer lugar, probar nuestro algoritmo con Big Data, especialmente con CNNs y datasets mucho más grandes, con objeto de confirmar los resultados obtenidos en este paper. Es de esperar que los resultados sean aún mejores, dado que el entrenamiento de la red sería aún más preciso. En segundo lugar, planteamos la modificación del Algoritmo 1, probando distintas configuraciones para el cálculo de σ_{fs}, tales como el uso de técnicas de ranking, o la eliminación de normalizaciones.

Referencias

[1] Y. Li, C.-Y. Chen y W. W. Wasserman, "Deep feature selection: theory and application to identify enhancers and promoters", *Journal of Computational Biology*, vol. 23, n.º 5, págs. 322-336, 2016.

Cuadro II

Resultados obtenidos. Dado que los datasets contienen características artificiales, sólo evaluamos su rendimiento hasta un número de características, como máximo, igual al número de características reales (válidas) de cada dataset.

Conjto.	# carac. válidas	Método	Mejor Precisión (Nº carac.)	Prec. 10 % carac. válidas	Prec. 25 % carac. válidas	Prec. 50 % carac. válidas	Prec. 100 % carac. válidas	AUC
Arcene	7000	MIM	81.0 (337)	75.0	74.0	74.0	73.0	0.736
		ReliefF	**83.0** (375)	**77.8**	**80.0**	**80.0**	71.0	**0.786**
		DeepLASSO	80.0 (464)	73.0	69.0	71.0	70.0	0.709
		Propuesta	81.0 (464)	76.0	72.0	77.0	**80.0**	0.773
Dexter	9947	MIM	**90.7** (103)	73.3	59.3	51.3	50.0	0.567
		ReliefF	86.0 (1204)	**85.0**	59.7	49.7	50.0	0.569
		DeepLASSO	87.3 (9)	62.3	53.0	49.7	50.0	0.533
		Propuesta	90.3 (28)	83.7	**77.0**	**72.0**	**52.0**	**0.732**
Dorothea	2500	MIM	94.3 (5)	88.0	81.7	63.1	**90.6**	0.794
		ReliefF	**94.6** (876)	**92.9**	**92.0**	23.4	**90.6**	0.682
		DeepLASSO	94.3 (5)	37.7	90.9	**90.3**	90.3	0.765
		Propuesta	94.3 (3)	90.9	89.7	88.9	89.1	**0.895**
Gisette	2500	MIM	98.4 (689)	97.0	98.1	**98.4**	97.9	0.978
		ReliefF	98.4 (1226)	97.7	98.0	**98.4**	**98.3**	0.980
		DeepLASSO	97.4 (82)	96.1	96.0	95.9	97.3	0.964
		Propuesta	**98.5** (516)	**98.0**	**98.5**	98.1	**98.3**	**0.981**
Madelon	2500	MIM	84.8 (13)	**67.8**	72.7	80.7	78.7	0.777
		ReliefF	85.3 (19)	62.7	61.8	80.7	**85.3**	0.754
		DeepLASSO	86.7 (6)	58.8	**86.0**	77.8	75.5	0.755
		Propuesta	**87.0** (6)	59.5	85.8	**86.2**	**85.3**	**0.828**

Cuadro III

Ranking de los algoritmos en cuanto a precisión, y en cuanto al número más bajo de características usadas (entre paréntesis)

Método	Posición Arcene	Pos. Dexter	Pos. Dorothea	Pos. Gisette	Pos. Madelon	Posición Media
ReliefF	1 (2)	4 (4)	2 (4)	2 (4)	3 (3)	2,4 (3,4)
Propuesta	2 (3)	2 (2)	2 (1)	1 (2)	1 (1)	**1,6** (1,8)
MIM	2 (1)	1 (3)	2 (3)	2 (3)	4 (2)	2,2 (2,4)
DeepLasso	3 (3)	3 (1)	2 (2)	3 (1)	2 (1)	2,6 (**1,6**)

[2] K. Simonyan, A. Vedaldi y A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps", *arXiv preprint arXiv:1312.6034*, 2013.

[3] A. Mahendran y A. Vedaldi, "Understanding deep image representations by inverting them", en *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, IEEE, 2015, págs. 5188-5196.

[4] R. Zhao, W. Ouyang, H. Li y X. Wang, "Saliency detection by multi-context deep learning", en *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, págs. 1265-1274.

[5] D. Zhang, D. Meng y J. Han, "Co-saliency detection via a self-paced multiple-instance learning framework", *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, n.º 5, págs. 865-878, 2017.

[6] V. Mnih, N. Heess, A. Graves y col., "Recurrent models of visual attention", en *Advances in neural information processing systems*, 2014, págs. 2204-2212.

[7] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel e Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention", en *International Conference on Machine Learning*, 2015, págs. 2048-2057.

[8] V. Bolón-Canedo, N. Sánchez-Maroño y A. Alonso-Betanzos, "A review of feature selection methods on synthetic data", *Knowledge and information systems*, vol. 34, n.º 3, págs. 483-519, 2013.

[9] S. Ioffe y C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", *arXiv preprint arXiv:1502.03167*, 2015.

[10] D. P. Kingma y J. Ba, "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*, 2014.

[11] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard y col., "TensorFlow: A System for Large-Scale Machine Learning.", en *OSDI*, vol. 16, 2016, págs. 265-283.

[12] M. A. Hall y L. A. Smith, "Practical feature subset selection for machine learning", 1998.

[13] I. Kononenko, "Estimating attributes: analysis and extensions of RELIEF", en *European conference on machine learning*, Springer, 1994, págs. 171-182.

[14] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann e I. H. Witten, "The WEKA data mining software: an update", *ACM SIGKDD explorations newsletter*, vol. 11, n.º 1, págs. 10-18, 2009.