**Función de activación softplus parametrizada para redes neuronales convolucionales profundas**

**Resumen**

Presentamos, en este trabajo, una nueva familia de funciones de activación dependiente de un parámetro que extiende la función de activación *softplus* (Dugas y col. 2000) y que denominaremos *softplus* parametrizadas, s+p de forma tal que acelere el aprendizaje utilizado en redes neuronales convolucionales y que empíricamente mostraremos que mejora la precisión del clasificador. Analizaremos sus propiedades, mostrando la sencillez del calculo de sus derivadas, potenciando que las derivadas sucesivas utilizadas en el algoritmo de retropropagación del error no se difuminen a través del gran número de capas ocultas del modelo de red neuronal. Al igual que la "unidad lineal exponencial" (ELU) (Clevert y col. 2016) (una ampliación de las unidades lineales rectificadas (ReLU), las ReLUs permeables (LReLU) y las ReLUs (PReLUs) parametrizadas), las funciones s+p tienen valores negativos suavizados que les permiten promover las activaciones de unidades cuyas medias estén más cerca de cero, como la normalización por lotes, pero con una menor complejidad computacional. Para mostrar el rendimiento de los modelos de red entrenados con las funciones s+p utilizaremos test de hipótesis no paramétricos asociados a diseños experimentales de tipo *holdout* para contrastar la bondad de nuestras funciones de activación frente a las funciones (ReLU, LReLU, PReLU, ELU, s+ y s+p). Utilizando modelos estándar de redes neuronales convolucionales se entrenaron los conjuntos de datos de visión por computador, CIFAR-100 y CIFAR-10. Los resultados mostraron que una vez determinado el valor del parámetro de la función utilizando una validación cruzada sobre el conjunto de entrenamiento, la función s+p aceleró el aprendizaje en estos dos conjuntos de datos de forma significativa (para un nivel de significación α=0,05) con respecto al resto de funciones de activación. También la función s+p mejoró de forma significativa, para α=0,05, el porcentaje de patrones bien clasificados y la mínima sensibilidad del clasificador con respecto al resto de funciones utilizando el test de Friedman.

**Palabras clave**: Funciones de activación, funciones s+p, normalización por lotes, aprendizaje profundo, redes neuronales convolucionales, test de Friedman, test de Holm

**Introducción**

En el trabajo vamos a analizar la relación que tienen las funciones de activación y la normalización por lotes en el entrenamiento de redes profundas (ver el trabajo de revisión de Bengio et al. (2009)). Este concepto de redes profundas suele estar asociado a redes neuronales donde el número de capas ocultas es tres o más.

Con respecto a las funciones de activación, en redes superficiales (en general aquellas que tienen, por tanto, menos de tres capas ocultas) las funciones de transferencia que nos interesan son funciones no lineales, estrictamente crecientes, continuas y derivables. Entre ellas son habituales las sigmoides o las arco tangente hiperbólica, dado que sus sencillas derivadas son fáciles de implementar en el algoritmo de retropropagación del error y donde dado el reducido número de capas ocultas el gradiente no se desvanece. En la redes profundas estas funciones de activación son más simples de forma tal que la derivadas sucesivas, (en este caso su número es mucho más alto) utilizadas en el algoritmo de retropropagación del error no se desvanezcan. De todas las propuestas hechas hasta ahora la función ReLU (Glorot et al. 2011), es la más utilizada, dado que su derivada es la unidad en el dominio positivo de la variable y cero en el dominio negativo. Esta sencillez, contrasta con el hecho de que en el dominio negativo de la variable no existe señal y que si primero se normaliza la salida de una capa oculta y luego se utiliza la función ReLU, la normalización desaparece.

Existen múltiples propuestas de funciones de activación en modelos de redes neuronales superficiales, de entre ellas, las neuronas con funciones de activación sigmoideas están más cercanas al concepto biológico de activación de una neurona, con las funciones de tipo tangente hiperbólica, en cambio, se obtienen mejores resultados para el entrenamiento de redes neuronales multicapa.

Para redes neuronales profundas las funciones de activación presentan características diferentes. Así, las funciones de activación (ReLU, LReLU, PReLU, ELU, s+ y s+p) han sido utilizadas

Las redes neuronales convolucionales RNC son, simplemente, redes neuronales multicapa en las que algunas capas realizan una operación de convolución en lugar de la

tradicional multiplicación matricial de entradas por pesos. Así, son redes neuronales artificiales profundas que se utilizan habitualmente para resolver múltiples problemas prácticos que requieren procesar señales, imágenes, fonemas, series temporales, etc. Se diferencian de las redes estándar en que tanto sus entradas como sus salidas pueden ser estructuradas. Las redes RNC nos permitirán aprovechar dicha estructura para diseñar arquitecturas especializadas para los diferentes problemas a abordar.

Las aplicaciones de RNC en análisis de imágenes médicas se remontan a la década de los 90, cuando se utilizaron para la detección asistida por computadora de nódulos pulmonares en conjuntos de datos CT [15]. Más recientemente esta habiendo una eclosión de trabajos en este campo donde citaremos como ejemplo los trabajos de detección automática de pólipos en videos de colonoscopia [17], o la detección asistida por computadora de embolia pulmonar (PE) en conjuntos de datos CT [18],

**Estado del arte**

En Glorot et al. 2011, los autores proponen explorar el uso de funciones de rectificación no lineal como alternativas a la tangente hiperbólica o sigmoide en redes neuronales artificiales profundas, además de usar un regularizador L1 en los valores de activación para promover la dispersión y prevenir posibles problemas numéricos con la activación ilimitada. Los autores afirman que esta función permite a las redes profundas lograr su mejor rendimiento sin utilizar pre entrenamiento no supervisado.

La función de rectificación lineal, Rectified Linear Units, ReLU, (Krizhevsky et al. 2012; Glorot et al. 2011), es de la forma y = frelu(x)= max(0,x). La derivada de esta función coincide con la función escalón utilizada por las neuronas de McCulloch y Pitts. Es una función asimétrica puesto que la respuesta a un patrón de entrada inhibidor es 0, esto es, no hay respuesta.

Como ventaja esta función permite que una red obtenga fácilmente representaciones dispersas. Por ejemplo, si hacemos una inicialización uniforme de los pesos, por ejemplo entre [-1 y 1] o una Normal (0,1), alrededor del 50% de los valores de salida continuos de las unidades ocultas son ceros y este porcentaje puede aumentar fácilmente con la regularización inducida por la dispersión. Por tanto, para una entrada dada, solo un subconjunto de neuronas estará activo, y la computación será lineal en este subconjunto, una vez que se selecciona este subconjunto de neuronas, la salida es una función lineal de la entrada. Podemos ver el modelo como un número exponencial de modelos lineales que comparten parámetros (Nair y Hinton, 2010). Debido a esta linealidad, los gradientes se calculan mucho más fácilmente puesto que no es necesario calcular la función exponencial en las activaciones y se puede explotar la dispersión.

La dispersión se ha convertido en un concepto de interés, no solo en neurociencia computacional y aprendizaje automático, sino también en estadística y procesamiento de señales (Candes y Tao, 2005). En redes profundas asociadas a autoencoders (Ranzato et al., 2007, 2008; Mairal et al., 2009) en redes profundas de creencia *Deep Belief Networks* (Lee et al., 2008).

Sin embargo, forzar demasiada dispersión puede perjudicar el rendimiento predictivo para un número igual de neuronas, ya que reduce la capacidad efectiva del modelo. De esta forma utilizar la función ReLU también tiene sus inconvenientes puesto que la fuerte saturación en 0 puede dañar la optimización al bloquear la retro-propagación del gradiente.

La función ReLU también presenta inconvenientes. Por ejemplo, algunos investigadores argumentan que la pendiente cero evita el aprendizaje de valores negativos (Maas et al., 2013; He et al., 2016b). De esta forma, otras funciones de activación han sido propuestas en la última década, como la unidad lineal rectificadora de fuga (LReLU) (Maas et al., 2013), unidad lineal rectificadora paramétrica (PReLU) (He et al., 2016b) y unidad lineal exponencial (ELU) (Clevert et al., 2015). Desafortunadamente, no hay consenso acerca de cómo estas nuevas propuestas mejoran los resultados en tiempo de computación y en rendimiento del clasificador comparadas con ReLU, y por lo tanto sigue siendo ReLU la función de activación más utilizada en aprendizaje profundo.

Otro tipo de funciones de activación han sido propuestas (Dugas et al., 2001), donde proponen la función de activación *softplus*(x)= s+= log (1+ex) para evaluar el impacto potencial de este efecto. Esta función es una versión suavizada de la función ReLU, de forma tal que en x= 0 la función pasa a ser continua y derivable. De hecho su derivada es la función sigmoide.

Existen otras variantes de las unidades lineales rectificadas, algunas utilizan una pendiente no nula α para valores negativos de x, así la primera es y = fret (z) = max(0, x) + α min(0, x), donde la rectificación en valor absoluto (Jarrett et al. 2009) utiliza α= −1 y se emplea en situaciones en las que no importa la polaridad de la señal de entrada, y = faret (z)= max(0,x)-min(0,x)= |x|. En estas situaciones no importa la polaridad de la señal de entrada, como en el reconocimiento de objetos en imágenes. De este tipo es la función ReLU con pérdidas (Maas et al 2013), en la que se utiliza un pequeño valor de α para evitar que la derivada sea siempre cero cuando x es negativo.

Una variante de este tipo de función es la función ReLU paramétrica, o PReLU donde la pendiente αj es un parámetro más de la neurona j-ésima, que hay que aprender

**Propuesta de función de transferencia**

En este trabajo proponemos una nueva función de activación a la que denominaremos s+2 extendiendo la función s+ al tercer cuadrante tratando de compatibilizar esta función con una normalización por lotes que mejore los resultados de clasificación asociados a las redes CNN.



en s+p establecemos un hiperparámetro α que define si la asíntota horizontal es s+2= 0 , s+p= -log2, o s+p= -1, y en general s+p= -α; manteniendo la otra asíntota en s+p=x-α (ver Figura 1). De esta forma podemos tener una respuesta pequeña o incluso nula a una entrada inhibitoria.



Figura 1. representación gráfica de la familia de funciones s+p para diferentes valores de α

α es un parámetro que deberemos de determinar para cada problema mediante una validación cruzada sobre el conjunto de entrenamiento, por ejemplo una partición en 5 subconjuntos disjuntos. La idea detrás del diseño s+p es aprender la mejor concavidad para las entradas negativas y positivas.

s+p tiene varias propiedades matemáticas que la relacionan con la función logística σ

1) Su derivada es la función sigmoide



La derivada de la función de activación decrece asintóticamente a 1 lo que la hace más interesante a la hora de evitar los problemas asociados a la saturación de esta función de activación, lo que permite que el algoritmo de entrenamiento de la red converja más rápidamente, al evitar la saturación prematura.

2) 

Esto nos permite calcular las derivadas segundas de s+2  con sólo dos sencillas operaciones una resta y una multiplicación dado. Con lo cual a diferencia de la funciones alternativas ReLU, LReLU, PReLU, ELU, SReLU, su segunda derivada no desaparece y es fácil de calcular; esto nos permitirá utilizar métodos de optimización utilizando la matriz Hessiana.

3) Presenta de forma compacta la siguiente tautología:



4) Tiene una asíntota en y= x-α

 

**Modelos e Inicialización**

Como ejemplo particular de un modelo VGG, utilizamos la variante VGG con diecinueve capas (VGG-19). Para entrenar modelos con un número considerablemente diferente de capas, elegimos la arquitectura ResNet con activación previa, con cincuenta y seis capas (ResNet-56) y también la ResNet con preactivación con ciento diez capas (ResNet-110). Para los experimentos se utilizo la inicialización de Kaiming (He et al., 2016b).

**Entrenamiento y Regularización**

**Diseño experimental**

Para validar la efectividad de nuestra función de activación, se realizaron una serie de experimentos entrenando modelos de clasificación para dos conjuntos de datos de imágenes disponibles públicamente y que habitualmente se han utilizado para validar el rendimiento de diferentes funciones de activación en aprendizaje profundo. A menudo se utilizan para validar el rendimiento del selector de genes y generadores. Describiremos brevemente esta bases de datos. Los conjuntos de datos son CIFAR-10 y CIFAR-100 (Krizhevsky y Hinton, 2009). El conjunto de datos CIFAR-10 posee diez clases que contienen 6000 imágenes, de las cuales mediante un procedimiento de *holdout*, 5000 se utilizan para el conjunto de entrenamiento, y 1000 quedan para el conjunto de test. Cada ejemplo de entrenamiento es una imagen RGB de tamaño 32×32×3. El conjunto de datos de imágenes CIFAR-100 tiene 100 clases, que a su vez contienen 600 imágenes de ejemplo cada una. De estos, también mediante un procedimiento de *holdout,* 500 se utilizan para el conjunto de entrenamiento y 100 se emplean para el conjunto de test. Cada ejemplo es una imagen RGB de 32×32×3. En ambos conjuntos no consideramos conjuntos de validación, así, para ajustar los parámetros de los modelos de red y el hiper-parámetro α de la función s+p utilizaremos una validación cruzada sobre el conjunto de entrenamiento.

Para definir el valor del hiperparámetro α de s+p, para la validación cruzada hemos tomado los siguientes los valores {0.00, 0.20, 0.40, 0.60, 0.80, 1.00} donde el valor de α= 0.00 se corresponde con la función s+. En base a estos resultados experimentales, decidimos establecer α= para CIFAR-10 y α= para CIFAR-100 .

Hemos considerado en este trabajo diferentes funciones de activación recientemente propuestas para redes RNC con el propósito de comparar el rendimiento en calidad del clasificador. En este estudio comparativo, definimos un experimento como el entrenamiento de un modelo profundo, {VGG-19 (%), ResNet-56 (%), ResNet-110 (%)}, que usa una función de activación distinta {ReLU, LReLU, PReLU, ELU, s+ y s+p} en un conjunto de datos dado CIFAR-10 y CIFAR-100. Si no se menciona lo contrario, llevamos a cabo diez ejecuciones de cada experimento. Definimos un escenario como el conjunto de experimentos con respecto a todas las funciones de activación en un conjunto de datos específico usando un modelo particular.

La evaluación de los diferentes modelos ha sido realizada utilizando la tasa de correcta clasificación (CCR) o medida de precisión. Para evaluar la estabilidad de los métodos, se ha realizado 10 veces un procedimiento de *holdout* y entrenamos los modelos durante 100 épocas, ya que es suficiente para que se sature el valor del CCR.

En los experimentos se utilizo para el algoritmo de retropropagación del error una tasa de aprendizaje inicial de 0.1 y una disminución de la velocidad de aprendizaje de 0.02 con pasos en las épocas 60, 80 y 90 para los conjuntos de datos CIFAR-10 y CIFAR-100.

Los experimentos emplearon mini lotes de tamaño 128 y descenso de gradiente estocástico con la técnica de aceleración de Nesterov como método de optimización. El parámetro de momento se estableció en 0.9 y la disminución de su valor fue igual a 0.0005 en las épocas citadas anteriormente.

Todos los experimentos se han realizado sin utilizar la técnica de la regularización aleatoria, *dropout* (Srivastava et al., 2014) ya que los estudios más recientes han mostrado que, a pesar de mejorar el tiempo de entrenamiento, el *dropout* no proporciona contribuciones significativas al rendimiento global del modelo (Ioffe y Szegedy, 2015). Además, el *dropout* se ha convertido recientemente en una técnica restringida a las capas más externas de la arquitectura, las cuales están completamente conectadas, que a su vez están siendo menos utilizadas en arquitecturas más recientes, siendo reemplazadas por una capa de *pooling* promedio (He et al., 2016a; Huang et al., 2016; He et al. al., 2016c; Zagoruyko y Komodakis, 2016, 2017).

Por último indicar que para la implementación de la arquitecturas de las redes hemos utilizado ....

**Resultados y discusión**

**References**

X. Glorot, A. Bordes, Y. Bengio (2011). Deep sparse rectifier neural networks. En Geoffrey Gordon, David Dunson, y Miroslav Dudík, editores, Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, volume 15 of JMLR W&CP, Proceedings of Machine Learning Research, pp 315-323, Fort Lauderdale, FL, USA, 11-13 Apr 2011. PMLR. URL http://proceedings.mlr. press/v15/glorot11a.

Ch. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, y R. García. Incorporating second-order functional knowledge for better option pricing. En Proceedings of the 13th International Conference on Neural Information Processing Systems, NIPS’00, pages 451-457, Cambridge, MA, USA, 2000. MIT Press. URL https://goo.gl/doB3GN

K. Jarrett, K. Kavukcuoglu, Marc’Aurelio Ranzato, y Y. LeCun. What is the best multi-stage architecture for object recognition? En IEEE 12th International Conference on Computer Vision, ICCV 2009, Kyoto, Japan, 2009, pp 2146-2153, 2009. ISBN 978-1-4244-4420-5.

A. L. Maas, A. Y. Hannun, y A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. En ICML Workshop on Deep Learning for Audio, Speech and Language Processing, 2013. URL https://goo.gl/5x3Cj6.

K. He, X. Zhang, S. Ren, y J. Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. arXiv e-prints, 2015a. URL http://arxiv.org/abs/1502.01852

Nair, V. y Hinton, G. E. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. Proceedings of the 27th International Conference on Machine Learning, pp. 807-814.

Krizhevsky, A., Sutskever, I., y Hinton, G. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In Advances In Neural Information Processing Systems. doi:http://dx.doi.org/10.1016/j.protcy.2014.09.007

He, K., Zhang, X., Ren, S., y Sun, J. (2016b). Delving deep into rectifiers: Surpassing

human-level performance on Imagenet classification. In Proceedings of the IEEE International Conference on Computer Vision. Institute of Electrical and Electronics Engineers Inc. volume 11-18 pp. 1026-1034.

Bengio, Y. (2009). Learning deep architectures for AI. Foundations and Trends in Machine Learning, 2(1), 1-127.

Clevert D. A., Unterthiner T. y Hochreiter S. Fast and accurate deep network learning by exponential linear units (ELUS). ICLR 2016.

[15] S.-C. B. Lo et al., “Artificial convolution neural network techniques and applications for lung nodule detection,” IEEE Trans. Med. Imag., vol. 14, no. 4, pp. 711-718, Dec. 1995.

[17] N. Tajbakhsh, S. R. Gurudu, and J. Liang, “Automatic polyp detection in colonoscopy videos using an ensemble of convolutional neural networks,” in Proc. IEEE 12th Int. Symp. o Biomed. Imag., 2015, pp. 79–83.

[18] N. Tajbakhsh and J. Liang, “Computer-aided pulmonary embolism detection using a novel vessel-aligned multi-planar image representation and convolutional neural networks,” in Proc. MICCAI, 2015.

Tabla: Evaluación de rendimiento de las bases de datos CIFAR-10 y CIFAR-100 VGG-19 100 épocas. Medias y desviaciones típicas y p-valores de los test de Friedman y Holm

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Functions | CIFAR 10 | CIFAR-100 |  |  |
| ReLU |  |  |  |  |
| LReLU |  |  |  |  |
| PReLU |  |  |  |  |
| ELU |  |  |  |  |
| s+ |  |  |  |  |
| s+p |  |  |  |  |