(Un)Confused Terminator

Luis Andrés Barrantes Ballestero
Escuela de Computación
Ingeniería en computación
Instituto Tecnológico de Costa Rica
Cartago, Costa Rica 18/05/2018
Email: http://www.machobarbal@gmail.com

Danny Chaves Chaves
Escuela de Computación
Ingeniería en computación
Instituto Tecnológico de Costa Rica
Cartago, Costa Rica 18/05/2018
Email: dxnnx@ieee.org

***Resumen*—Este artículo trata sobre la utilización de redes neuronales multicapa programada desde 0 para resolver problemas de reconocimientos de números sin importar el fuente de estos, dichos números son blancos con el fondo negro. Se usó el set de datos de NMIST el cual cuenta con 60 mil imágenes de una dimensionalidad de 28x28, este set de datos fue el mismo utilizado por YanLecun para resolver este problema en 1989.[2]**
Se probó una red neuronal con distintos números de neuronas y capas ocultas para lograr un mejor porcentaje de aciertos el cual logro ser de 97.98 % con 4 capas de 128,64,32 y 16 neuronas respectivamente, además cabe destacar que se utilizó ReLu y Sigmoid como función de activación y softmax con cross-entropy para calcular la pérdida de nuestra red. Se hicieron pruebas con números propios además del set de datos los cuales dieron resultados muy positivos de un porcentaje de aciertos de 98 %. En general este artículo dio resultados positivos en comparación a otros métodos utilizados en el sitio de LeCun, sin embargo, no hay un gran cambio con respecto a lo esperado sobre los experimentos realizados, ya que estos varían no más de un 3 % sin importar el tamaño de la capa o la profundidad de la red.

I. Introducción

Las redes neuronales a día de hoy es uno de los métodos más utilizados para resolver problemas de IA, por ello se creará una red neuronal en Python, utilizando únicamente la librería numpy como apoyo en las operaciones matemáticas, esto para resolver el problema de identificar números escritos por cualquier persona o máquina sin importar el fuente, su único requisito es estar en fondo negro con el número en blanco. para ello se usará una red neuronal con la cual se experimentará diversos hiper-parámetros para lograr un porcentaje de acierto mayor. Se utilizará el set de datos de NMIST que fue el mismo que utilizó Yann LeCun cuando resolvió este problema por primera vez, este set de datos cuenta con 60 mil imágenes de 28x28[1]
Esta red neuronal será de perceptrones multicapa, se harán diversos experimentos sobre los tamaños de cada capa, se usará cross-entropy para calcular la perdida además de ReLu como función de activación.
Se utilizó como base del código el proyecto de Ritchie Vink, el cuál es una red con muchas de las funciones necesitadas.[2]

II. Trabajo Relacionado

Respecto a este problema existe muchos proyectos y artículos que hablan de ello, los cuales utilizan diferentes métodos para resolver el problema de identificar números de una foto entre ellos destacan modelos lineales simples que logran un porcentaje de error menor a 12 %, otros modelos que usan KNN que logran un error menor al 5 %, los cuales son bastante buenos para ser métodos tan simples en el aspecto de IA, sin embargo, estos modelos no tienen nada en comparación con modelos hechos de redes de convolución que cuentan con alrededor de 6 capas, dicho modelo cuenta con un porcentaje de error menor al 0.3 %.
Todos estos experimentos se encuentran en el sitio de Yan LeCun, ya que este problema es considerado como una buena práctica para comenzar a trabajar en IA.

III. Metodología

Para resolver este problema se optó por desarrollar una red neuronal multicapa con la cuál se harán experimentos sobre la cantidad de capas y los tamaños de estas para lograr un mejor porcentaje de aciertos. Para está red se usará ReLu como función de activación entre cada capa y cross-entropy con softmax para calcular el loss que tiene nuestra red.

III-A. Red Neuronal-Perceptrón Multicapa(MLP)

Brevemente se puede definir este tipo de red como una red neuronal artificial formada por múltiples capas, esto le permite resolver problemas que no son linealmente separables con ayuda de una función de activación, lo cual es la principal limitación del perceptrón, en la figura 1 se puede ver como funciona un perceptrón.[3]c

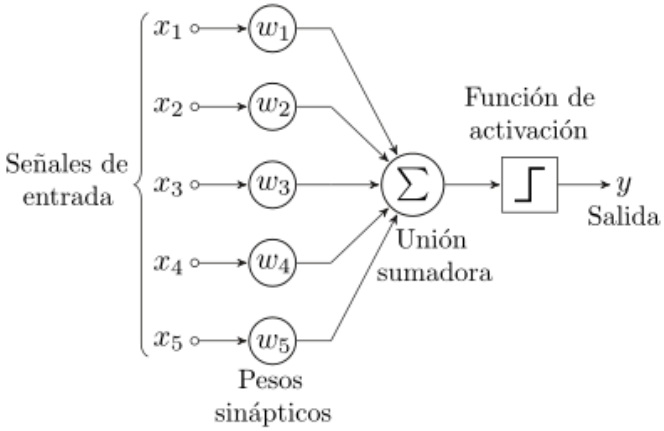

Figura 1. Ejemplo deL modelo de un perceptrón[3]

El perceptrón multicapa puede ser totalmente o localmente conectado. En el primer caso cada salida de una neurona de la capa "i" es entrada de todas las neuronas de la capa "i+1", mientras que en el segundo cada neurona de la capa "i" es entrada de una serie de neuronas (región) de la capa "i+1" como se logra observar en la figura 2. Para este experimento se conectará completamente una capa con la otra.[3]

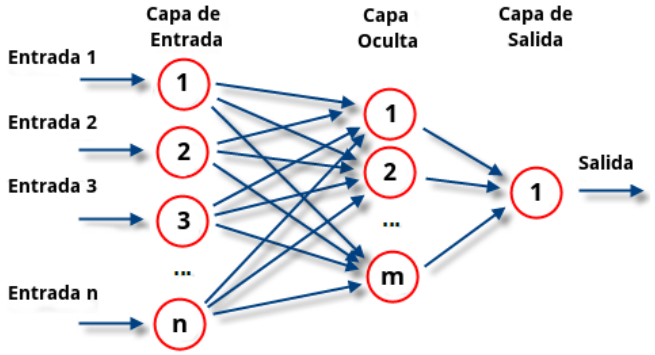

Figura 2. Conexiones del perceptrón multicapa[3]

Estas pueden ser de entrada, la cual en el caso de este problema son las imágenes de entrada, las capas ocultas que son capas de procesamiento intermedio y la capa de salida, que para este problema tendrá 10 neuronas, las cuales cada una se identificará con una clase.
Vale la pena para este problema recordar las limitaciones de este tipo de red, la cual si se entrena mal no logra dar resultados aceptables, además de que puede darse que en la función de error encuentre solamente mínimos locales lo cuál le impedirá converger. Por ello se probará con distintos números de capas ocultas y neuronas en ellas para así evitar este problema lo máximo posible.[3]

III-B. SoftMax y Cross Entropy

La función Softmax toma un vector N-dimensional de números reales y lo transforma en un vector de número real en el rango (0,1) Se define como:

$$Pj = \frac{e^{ai}}{\sum_{k=1}^{N} e_k^a}$$

Donde K son las posibles salidas y a es el valor real en el vector.
Como su nombre lo sugiere, la función Softmax es una versión "suave"de la función máxima. En lugar de seleccionar un valor máximo, selecciona un todo (1) con el elemento máximo obteniendo la porción más grande de la distribución, pero otros elementos más pequeños obtienen parte de él también, lo cual es útil para no menospreciar datos importantes.[4]
Esta propiedad de la función Softmax que genera una distribución de probabilidad la hace adecuada para la interpretación probabilística en tareas de clasificación. Como su nombre lo sugiere, la función Softmax es una versión "suave"de la función de maximización. En lugar de seleccionar un valor máximo, rompe el todo (1) con el elemento máximo obteniendo la porción más grande de la distribución, pero otros elementos más pequeños obtienen parte de él también.[4]
EL Cross-Entropy indica la distancia entre lo que el modelo cree que debe ser la respuesta y cuál es realmente la respuesta original. Se define como:

$$H(y,p) = \sum_{yi} \log(pi)$$

Donde Y es la etiqueta o resultado de la imagen y p es el resultado de hacerle SoftMax a la imagen o entrada de la neurona x.[4]
Cross-Entropy es una alternativa ampliamente utilizada al MSE. Se utiliza cuando se puede entender que las activaciones de nodo representan la probabilidad de que cada hipótesis sea verdadera, es decir, cuando la salida es una distribución de probabilidad. Por lo tanto, se utiliza como una función de pérdida en redes neuronales que tienen activaciones de Softmax en la capa de salida.[4]
Para poder hacer backprop en nuestro modelo necesitaremos la derivada de la pérdida Cross-Entropy con Softmax y está es:

$$\frac{\delta L}{\delta O_i} = (p_i - y_i)$$

donde p este el resultado de evaluar x con SoftMax y y es la etiqueta con el resultado de la imagen o entrada de la neurona x. Este cálculo se encuentra en el artículo de E.Bendersky.[5]

III-C. ReLu

En este experimento se utilizará como función de activación ReLu, el cuál simplemente parte en 2 una función lineal, convirtiéndolo en una no linealidad, esto ayuda a los perceptrones a eliminar su desventaja de servir en problemas únicamente lineales.[6] La función de ReLu está dada por :

$$f(x) = \text{máx}(O, X)$$

Donde x es la entrada de la neurona.
Se utiliza ReLu debido a que es una manera simple de activar las neuronas de un problema sencillo como este, como se explicó anteriormente con ReLu cada perceptrón al final será activado con esta función, donde su idea es crear no linealidades para poder resolver problemas no lineares como sería el caso de este ejemplo, donde es muy posible que un sistema linear se quede corto.[6]

IV. Experimentos

Los experimentos se dividirán en 3 grandes tipos, sin embargo, se harán pruebas previas como definir el tamaño de los batches, cambiar el learning rate e incluso probar el modelo utilizando MSE. Cabe destacar que se probará tanto como con el set de datos de Mnist como con un set de datos propio . Las pruebas principales serán con Cross-Entropy/Softmax. Además se mostrará por cada uno de los experimentos en el mejor caso como fue cambiando el loss y el acurracy.

IV-A. Tamaños diversos de capas ocultas

Se probará con los siguientes tamaños de capas 128,256,512,1024, además, se espera que los resultados sean variantes, apostando a que las capas más pequeñas darán mejores resultados debido a que la cantidad de parámetros necesarios en la red se intuye será pequeña. Se harán pruebas con una y dos capas, sin embargo el segundo experimento es el que contrastará la diferencia entre estos resultados. Se espera que no cambie demasiado el porcentaje de aciertos, donde las redes más pequeñas puedan dar cerca de un 80 % y las más grandes un 70 %.

IV-B. una capa oculta vs dos capas ocultas

Se probará la red con una y con dos capas ocultas utilizando los tamaños del experimento anterior. Se espera que entre mayor sea el número de capas mejor será el resultado, donde es posible que con dos capas el porcentaje de aciertos sea cercano al 80 % y con una sola pueda disminuir a un 50 %.

IV-C. Varias capas ocultas muy pequeñas

Siguiendo un poco la intuición de que no serán necesarios muchos parámetros pero si la identificación de muchos puntos claves de cada imagen se probará crear una red de 3 a 4 capas con una cantidad de 128 o menos neuronas en cada capa, se espera que los resultados sean superiores al 90 % en este caso.

V. Resultados

Se mostrarán los resultados obtenidos de las pruebas previas en primera instancia, seguido de los resultados de los tres experimentos.

Al usar MSE dio resultados con una capa oculta de 256 de tamaño en el set de datos de MNIST dio como resultado un loss de 0.2 y un porcentaje de acierto de un 30 %, con el set propio dio como resultado un loss de 0.1 y un porcentaje de acierto de 20 % .
Con una capa oculta de 128 de tamaño en el set de datos de MNIST dio como resultado un loss de 0.29 y un porcentaje de acierto de un 23 %, con el set propio dio un loss de 0.22 y un porcentaje de acierto de 20 % .
Con dos capas ocultas, una de 256 y otra de 128 de tamaño en el set de datos de MNIST dio como resultado un loss de 0.1 y un porcentaje de acierto de un 84 %, con el set propio dio un loss de 0.14 y un porcentaje de acierto de 80 % .
Con dos capas ocultas, una de 128 y otra de 64 de tamaño en el set de datos de MNIST dio como resultado un loss de 0.13 y un porcentaje de acierto de un 85 %, con el set propio dio un loss de 0.13 y un porcentaje de acierto de 80 % .
Al usar batches de 16,32 y 64 con dos capas ocultas, una de 128 y otra de 64 utilizando Cross-Entropy. Los datos fueron muy similares, variando solo un 2 % siendo con un batch de 16 imágenes un 94 % de aciertos, con 32 un 95 % y con 64 un 94 %.
Al usar el learning rate recomendado de un 0.0085 con dos capas ocultas una de 128 y otra de 64, usando Cross-EntropySotmax dió un 95 % de aciertos y un 0.09 % el loss.
Al aumentar el learning rate del problema de un 0.0085 a un 0.02 los resultados dan un 91 % de aciertos y un 0.11 % el loss. El disminuir el learning rate a un 0.0015 los resultados dan un 93
Dado estos datos para las siguientes pruebas se usarán batches de 32 y un learning rate de 0.0085.

V-A. Tamaños diversos de capas ocultas, con una o dos capas

Cabe destacar que se usa Cross-Entropy con Softmax. Se mostrarán dos tablas, en el Cuadro I son pruebas con una sola capa y el Cuadro II son pruebas con dos capas, cada tabla tendrá tamaños que varían de 1024 a 128.

Cuadro I
Con una capa

Primera capa	Loss	Aciertos
1024	0.1405	94.77 %
512	0.1304	95.80 %
256	0.1402	94.63 %
128	0.128	95.41 %

Se muestra en el caso con mejores resultados como que el loss fue disminuyendo a lo largo del entrenamiento y el Acurracy va aumentando, esto en el caso número 2 de la tabla anterior.

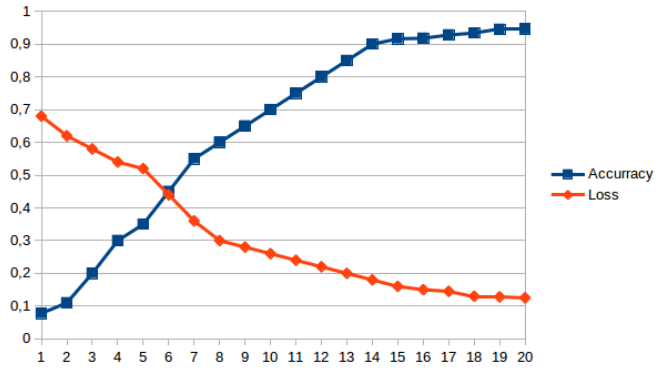

Figura 3. Se muestra como baja el loss y aumenta el acurrracy con una capa

Cuadro II
CON DOS CAPAS

Primera capa	Segunda Capa	Loss	Aciertos
1024	512	0.124	96.69 %
512	256	0.118	97.20 %
256	128	0.116	97.55 %
128	64	0.114	97.76 %

Se muestra en el caso con mejores resultados como que el loss fue disminuyendo a lo largo del entrenamiento y el Acurracy va aumentando, esto en el caso número 4 en la tabla anterior.

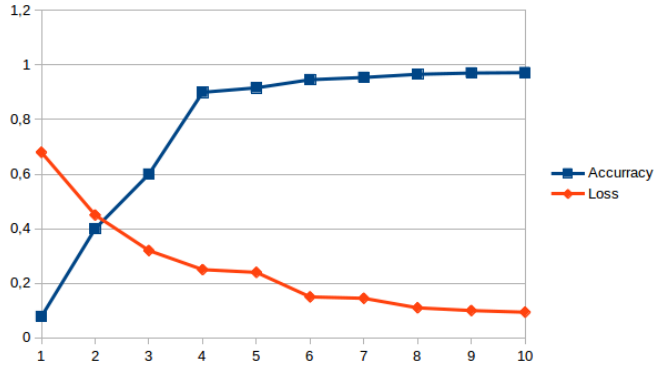

Figura 4. Se muestra como baja el loss y aumenta el acurrracy con dos capas

V-B. Varias capas ocultas muy pequeñas

Pruebas con 4 capas ocultas de 128,64,32 y 16 de tamaño da un 97.98 % de aciertos. Pruebas con 4 capas ocultas de 64, 64,32 y 32 de tamaño da un 97.45 % de aciertos.
Con 3 capas ocultas de 64,64 y 32 da un 97.84 % de aciertos.Con 3 capas ocultas de 128,64 y 32 da un 97.24 % de aciertos.
Se muestra en el caso con mejores resultados como que el loss fue disminuyendo a lo largo del entrenamiento y el Acurracy va aumentando, esto en el primer caso mencionado en esta sección.

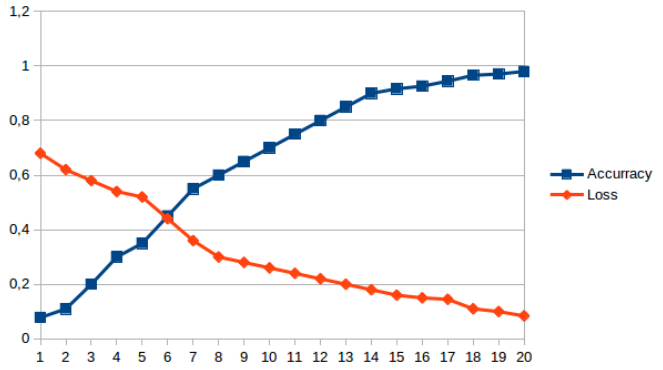

Figura 5. Se muestra como baja el loss y aumenta el acurrracy con cuatro capas

VI. CONCLUSIÓN

Este proyecto dio muy buenos resultados, mucho mejores que los esperados, sin embargo, no fueron muy variados unos de otros, ya que se esperaba que al agregar capas o al cambiar la cantidad de neuronas el porcentaje de aciertos aumentaría bastante pero fue menor a un 3 % aunque si se demuestra que con una sola capa si da resultados un poco peores que los de dos o mas capas, además que con capas pequeñas suele dar un mejor resultado aunque poco perceptible. Los experimentos extra de varias capas dieron resultados un poco mejores a las pruebas de 2 capas, sin embargo, como se mencionó anteriormente no cambiaron demasiado. También se observó como durante el entrenamiento la pérdida fue disminuyendo, así como el porcentaje de aciertos aumentó considerablemente. Se podría concluir que en este tipo de problemas aumenta levemente los aciertos. Además se probó que con MSE da resultados muy variados, desde una precisión muy baja hasta un considerable 80 % a pesar de ser tan sencilla para este problema.

VII. TRABAJO A FUTURO

Como trabajo a futuro podría ser bastante útil conseguir un set de datos que no siga el formato de fondo negro y el número en blanco, además de crear una red de convolución previa a las capas de la red, de esta manera puede reducir un poco la dimensionalidad de los datos y los parámetros que tenga la red en total, aplicando diversos filtros, con el fin de que el programa pueda incluso leer números con distinto formato, además de aumentar su velocidad de entrenamiento y posiblemente su porcentaje de acierto.

REFERENCIAS

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. 1998 ”Gradient-based learning applied to document recognition.”Proceedings of the IEEE, 86(11):2278-2324, November 1998. Available at: http://yann.lecun.com/exdb/mnist/ Accessed 5 May 2018.
[2] ritchie46/vanilla-machine-learning”, GitHub, 2017.Online. Available: https://github.com/ritchie46/vanilla-machine-learning/blob/master/vanilla_mlp.py. Accessed: 27- Apr- 2018.
[3] El perceptrón y perceptrón multicapa ¿Qué es y con que se come?”, El Blog de Sinfallas, 2018. Available: https://sinfallas.wordpress.com/2017/11/14/el-perceptron-y-perceptron-multicapa-que-es-y-con-que-se-come/. Accessed: 12- May- 2018.

[4] P. Dahal, Çlassification and Loss Evaluation - Softmax and Cross Entropy Loss", DeepNotes, 2018. [Online]. Available: https://deepnotes.io/softmax-crossentropy. [Accessed: 06- May- 2018].
[5] E. Bendersky, "The Softmax function and its derivative - Eli Bendersky's website", Eli.thegreenplace.net, 2016. Online. Available: https://eli.thegreenplace.net/2016/the-softmax-function-and-its-derivative/. Accessed: 1- May- 2018.
[6] D. Becker, Rectified Linear Units (ReLU) in Deep Learning — Kaggle", Kaggle.com, 2016. Online. Available: https://www.kaggle.com/dansbecker/rectified-linear-units-relu-in-deep-learning. Accessed: 05- May- 2018.
[7] Pdfs.semanticscholar.org, 2014. Online. Available: https://pdfs.semanticscholar.org/f287/51818854d9dbb27e60706c4161b40a326d81.pdf.Accessed: 12- May- 2018.