Guion

[n] hará referencia a la diapositiva n.

El problema que se estudiará es la optimización en el aprendizaje de máquina [1].

Tenemos un problema de clasificación que resolveremos con una red neuronal. La red tiene un número de capas ocultas **K**, cada una de ellas con sus nodos correspondientes, y en los extremos la entrada y salida de la red. La salida consiste en un vector que clasifica la entrada según un número de categorías. [2]

Para resolver este problema de clasificación se busca hallar los valores de cada entrada de $\mathbf{W^i}$, matriz de pesos para cada capa **i**, de forma que clasifique de la mejor forma posible las posibles entradas que podamos recibir. Estas matrices son importantes puesto que están involucradas en la transición de cada capa y en el cálculo de la activación de cada neurona [3].

Generalmente, para la resolución del problema de clasificación buscamos minimizar una función de costo cuyos parámetros tiene a cada $\mathbf{W^i}$. El articulo define el problema de optimización como lo siguiente. [4]

$$\min_{\{W^k\}_{k=1}^K} \ell(Y, \Phi(X, W^1, \ldots, W^K)) + \lambda\Theta(W^1, \ldots, W^K),$$

En este caso ***l*** es la función de costo con parámetros: la salida de algoritmo que depende de cada $\mathbf{W^i}$ y la entrada **X,** y **Y** una matriz con sus filas siendo la respuesta esperada por la red neuronal para cada ejemplo del **conjunto de entrenamiento**. El otro término en la función a optimizar tiene relación con el problema de **overfitting**, que ocurre cuando la red clasifica bien según el **conjunto de entrenamiento**, pero no hubo mucho aprendizaje al haber un mal rendimiento en el **conjunto de prueba** [4].

La importancia de resolver este problema es evidente, cada problema de aprendizaje de maquina involucra una función de costo, y es necesario para su resolución conocer qué condiciones debemos imponer sobre los parámetros para garantizar una solución satisfactoria [5].

En el curso, vimos el algoritmo del perceptrón para datos separables. Allí el costo era dado por todos aquellos ejemplos mal clasificados. El mínimo era 0 si los datos eran separables, puesto que finalmente lográbamos dar con una buena clasificación al aplicar el algoritmo iterativamente [6].

Nos vamos a enfocar en el problema de la no convexidad para redes neuronales. Una función es convexa si al trazar el gráfico de la función y tomar dos puntos cualesquiera, la recta que une los dos puntos está por encima del gráfico [7].

Esta condición de convexidad es importante para minimizar funciones. Una razón es porque se puede demostrar que los mínimos locales (óptimos locales) son mínimos globales (óptimo global) [8].

Debido a que muchos de los algoritmos utilizados para la minimización de la función de costo, como descenso por gradiente, convergen a un punto crítico (el gradiente es cero), ocurre que en un problema con una función de costo no convexa no siempre podemos garantizar que ese punto al que se convergió sea el mínimo. Puede ser un punto de silla, o un mínimo local [9].

La función de costo puede no ser convexa porque su dominio no es convexo (toda recta que une dos puntos debe estar contenida en el conjunto) o es la composición de algunas funciones que no son necesariamente convexas, entre otras razones [10].

Por ejemplo, en el caso de tomar como función de costo la diferencia entre los resultados esperados y los conseguidos

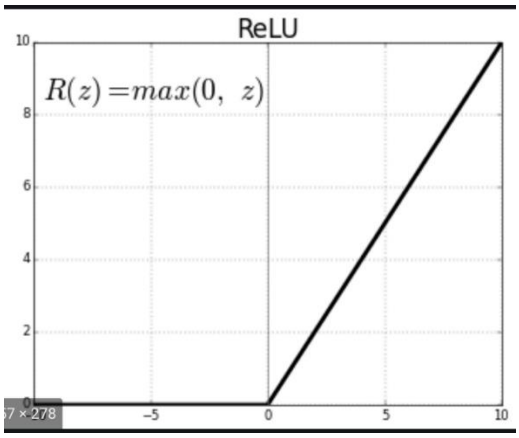
$$\ell(Y, \Phi) = \|Y - \Phi\|^2$$

, una función comúnmente convexa, pero hay casos donde no se logra un problema convexo por las funciones tomadas entre las transiciones entre las capas [11].

$$\Phi(X, W^1, \ldots, W^K) = \psi_K(\psi_{K-1}(\cdots \psi_2(\psi_1(XW^1)W^2)\cdots W^{K-1})W^K).$$

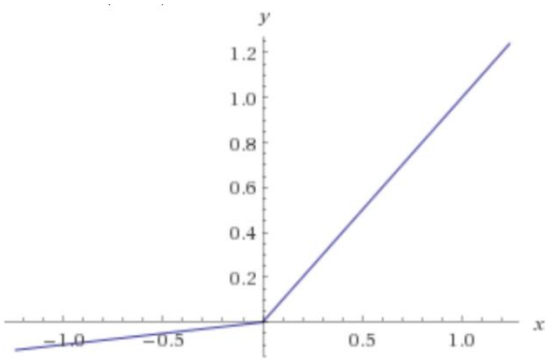
Una primera estrategia consiste en inicializar, todas las matrices de peso $\mathbf{W^i}$**,** de forma aleatoria. Luego se aplica algún algoritmo de optimización y se verifica qué tan rápido desciende el costo. Si no es lo suficientemente rápido, no puede ser un mínimo y se reinicia el algoritmo. Este proceso se ha visto funciona cuando el tamaño de la red neuronal es lo suficientemente grande [12].

Otra estrategia involucra la función **ReLU**. Al utilizarla como función de activación para las capas de la red neuronal ψ_i, dada una red neuronal suficientemente grande, da un rendimiento tan bueno como se necesite en el problema de clasificación [13].



Este buen rendimiento, se ha visto se relaciona con el hecho de que cierto número de neuronas no llegan a ser activadas, conocidas como **neuronas muertas**. No obstante, si el número de **neuronas muertas** es demasiado grande, puede llegar a afectar el rendimiento. Este fenómeno ocurre cuando para un **conjunto de entrenamiento,** la activación de una neurona es negativa para todo elemento del conjunto, cuando se aplica descenso por gradiente la neurona no será ajustada pues el gradiente será **0** por la definición de la función **ReLU** [13].

Para aprovechar el rendimiento de la función **ReLU**, pero combatir lo anteriormente mencionado, se utiliza una función similar a la **ReLU** conocida como **Leaky ReLU**. Donde el gradiente para valores menores o iguales a **0** no es cero y permite el ajuste de todas las neuronas que tendrán activaciones no muy distintas para las neuronas que antes estaban muertas [14].



De este problema surgen dos preguntas que requieren precisión matemática para explicar lo obtenido de forma experimental. La primera pregunta se relaciona con el tamaño de las redes neuronales, que pueden ser el número de neuronas por capa, o el número de capas. Se mencionó que, si este era suficientemente grande, al aplicar algunas funciones de activación

o algoritmos de optimización, se podía obtener un buen rendimiento en el problema no convexo. Pero, no se ha logrado dar con una respuesta precisa sobre un tamaño mínimo de la red neuronal dado un rendimiento pedido [15].

Esta anterior pregunta se relaciona con la función **ReLU**, ya que con un tamaño suficiente de una red neuronal la función **ReLU** clasifica tan bien como se requiera. Esto nos dice que, dado un tamaño acorde, sería posible que los mínimos locales fueran todos mínimos globales al alcanzar un rendimiento muy bueno, como ocurre en un problema convexo. Sin embargo, no existe una explicación matemática sobre este hecho ni tampoco se precisa mucho según lo que sería un tamaño acorde. Estas dos preguntas por resolver son claves a la hora de saber si el rendimiento pedido a una red neuronal es posible al fijar su tamaño y es dado un **conjunto de entrenamiento** [15].