UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERIA
DEPARTAMENTO DE INGENIERIA INFORMÁTICA

ANÁLISIS DE LA EFICIENCIA DEL ENTRENAMIENTO DE REDES NEURONALES PROFUNDAS BASADO EN SIMULATED ANNEALING

Felipe Alberto Reyes González

Profesor Guía: Victor Parada
Tesis de grado presentada en conformidad a los requisitos para obtener el grado de Magíster en Ingeniería Informática

Santiago, Chile
2017

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

CAPÍTULO 1. INTRODUCCIÓN

1.1 ANTECEDENTES Y MOTIVACIÓN

El aprendizaje profundo (*Deep learning*, DL) se refiere a una nueva clase de métodos de las máquinas de aprendizaje (*Machine learning*, ML). El proceso ocurre a través de muchas capas distribuidas en una arquitectura jerárquica que se puede utilizar para clasificar un patrón y el aprendizaje de características (G. E. Hinton et al., 2006; Bengio, 2009). Esta arquitectura se inspira en la inteligencia artificial que emula el proceso de aprendizaje profundo y en capas de las áreas sensoriales primarias del neocórtex en el cerebro humano, que extrae automáticamente rasgos y abstracciones de los datos (Bengio y LeCun, 2007; Bengio et al., 2013; Arel et al., 2010).

En los últimos años, se han desarrollado una serie de investigaciones en base a los algoritmos del DL en varios campos diferentes (LeCun et al., 2015). Ha sido utilizado para tareas de reconocimiento de imágenes (Krizhevsky et al., 2012; Farabet et al., 2013; Tompson et al., 2014; Szegedy et al., 2015) y de reconocimiento de voz (Mikolov et al., 2011; G. Hinton et al., 2012; Sainath et al., 2013), y han superado otras técnicas de aprendizaje en la predicción de la actividad de las moléculas de fármacos (Ma et al., 2015), en el análisis de datos en el acelerador de partículas (Ciodaro et al., 2012; Adam-Bourdarios et al., 2015), en la reconstrucción de los circuitos cerebrales (Helmstaedter et al., 2013), y en la predicción de los efectos de las mutaciones en el ADN no codificante en la expresión genética y en enfermedades (Leung et al., 2014; Xiong et al., 2015). También ha producido buenos resultados en diversas tareas para la comprensión del lenguaje natural (Collobert et al., 2011), en particular para la clasificación de temas, análisis de sentimientos, respuesta a preguntas (Bordes et al., 2014) y en la traducción (Jean et al., 2014; Sutskever et al., 2014).

Es el surgimiento del DL que ha permitido que las redes neuronales artificiales (*Artificial Neural Networks*, NN) sean nuevamente estudiadas (Bengio et al., 2006; G. E. Hinton et al., 2006; Le et al., 2012; Ranzato et al., 2007). A pesar de que el nuevo enfoque abarca diversos algoritmos (Bengio y LeCun, 2007; G. E. Hinton et al., 2006), un principio en común es que una NN con múltiples capas ocultas, que la convierten en profunda, puede codificar características más complejas en sus capas. Las NN fueron comunmente entrenadas a través del algoritmo

de retropropagación (Rumelhart et al., 1986b), que utiliza el método del gradiente estocástico descendente (*Stochastics descent gradiente*, SDG), o una de sus variantes, para actualizar los pesos de la NN y de esa manera reducir el error total. El problema durante el proceso de aprendizaje de las NN es descrito como un problema de minimización de una función de error, la que depende de los pesos que conforman la red (Rumelhart et al., 1986a). Este problema de optimización tiene la desventaja de ser no lineal, no convexo, además de tener mas de un mínimo local. Para solventar este problema se han desarrollado diversos algoritmos (Grippo, 1994; Jacobs, 1988; V. P. Plagianakos et al., 2002; Rumelhart et al., 1986b; V. Plagianakos et al., 1998) y su rendimiento varía según el problema a resolver. Por otra parte, su estructura le otorga la capacidad de aproximar cualquier función continua (Hornik, 1991), por lo tanto resuelven una amplia gama de problemas, como el reconocimiento de patrones (Jain et al., 2000), el agrupamiento y la clasificación (Zhang, 2000), la aproximación de funciones (Selmic y Lewis, 2002), la bioinformática (Mitra y Hayashi, 2006), procesamiento de señales (Hwang et al., 1997) y el procesamiento del habla (Gorin y Mammone, 1994), entre otros.

El enfoque clásico para el entrenar NN es la aplicación de algoritmos basados en el gradiente, como la retropropagación (Rumelhart et al., 1986b). El algoritmo de retropropagación busca minimizar la función de error mediante el uso del gradiente. Aunque la función de error disminuye rápidamente en la dirección del gradiente negativo, la retropropagación es generalmente ineficiente y poco fiable (Gori y Tesi, 1992) debido a la superficie de error. Además, su rendimiento se ve afectado por parámetros que deben ser especificados por el usuario, pues no existe una base teórica para escogerlos (Nguyen y Widrow, 1990). Dichos parámetros tienen una importancia crucial en el buen funcionamiento del algoritmo, por lo que el diseñador está obligado a seleccionar parámetros como los pesos iniciales de la NN, la topología de la red y la tasa de aprendizaje. En diversas investigaciones (Cauchy, 1847; Grippo, 1994; V. Plagianakos et al., 1998; V. P. Plagianakos et al., 2002) ha quedado demostrado que pequeñas modificaciones en estos valores influyen en el rendimiento de la NN.

Para proporcionar una convergencia más rápida y estable se han desarrollado diversas variaciones y alternativas a la retroprogación. Algunos de estos métodos modifican el valor de algún término durante el proceso (Jacobs, 1988; Rumelhart et al., 1986b) o de una tasa variable de aprendizaje (Jacobs, 1988; Vogl et al., 1988). Magoulas, Vrahatis, y Androulakis (1997); V. Plagianakos et al. (1998) propusieron dos técnicas para evaluar en forma dinámica la tasa de aprendizaje y las evaluaciones de gradiente. El primero se basó en el algoritmo de Barzilai y Borwein (Barzilai y Borwein, 1988) que adapta la tasa de aprendizaje; mientras que el

segundo utiliza estimaciones de la constante de Lipschitz, explotando la información local de la superficie de error y los pesos posteriores (Magoulas et al., 1997). Hay evidencias (Magoulas et al., 1997; V. P. Plagianakos et al., 2002; V. Plagianakos et al., 1998) que han demostrado que la retropropagación con algoritmos que adaptan la velocidad del aprendizaje son robustas y tienen un buen rendimiento para el entrenamiento de NN.

Se han sugerido diversos métodos para mejorar la eficiencia del proceso de minimización del error. Algunos de los métodos utilizados son métodos de segundo orden como el gradiente conjugado (Fletcher y Reeves, 1964; Hestenes y Stiefel, 1952; Polak E., 1969) y el quasi-Newton (Huang, 1970; Nocedal y Wright, 2006). Los métodos del gradiente conjugado utiliza una combinación lineal de la dirección de búsqueda anterior y el gradiente actual lo que produce una convergencia generalmente más rápida, es adecuado para redes neuronales de gran escala debido a su simplicidad, sus propiedades de convergencia y la poca memoria que requiere. En la literatura se encuentran diversos métodos basados en el gradiente conjugado (Birgin y Martínez, 2001; Møller, 1993) que han sido utilizados para la construcción de NN en varias aplicaciones (Charalambous, 1992; Peng y Magoulas, 2007; Sotiropoulos et al., 2002). Los métodos quasi-Newton se consideran como los algoritmos más sofisticados para el entrenamiendo rápido de una NN. Definen la dirección de búsqueda mediante una aproximación de la matriz Hessiana, requiriendo información adicional. Se han propuesto diversas estrategias para obtener una aproximación a la matriz Hessiana (Al-Baali, 1998; Nocedal y Yuan, 1993; S. Oren, 1972; S. S. Oren y Luenberger, 1974; Yin y Du, 2007); estas estrategias combinadas con búsquedas lineales han permitido definir una convergencia superlineal (Yin y Du, 2007), mejorando significativamente el rendimiento de los métodos originales. Otras propuestas han utilizado capas de preentrenamiento (G. E. Hinton y Salakhutdinov, 2006).

Debido a que la retropropagación suele minimizar una función de error no convexa con técnicas determinísticas locales (métodos de gradiente de primer y segundo orden), la probabilidad de converger a un mínimo local es alta (Bianchini y Gori, 1996; Bishop, 1995; Battiti y Tecchiolli, 1995). Este y otros problemas asociados con la retropropagación, como la convergencia lenta y la sobreejecución, podrían hacer que el proceso de aprendizaje fuera ineficiente. Para hacer frente a estos problemas han surgido técnicas para solucionarlo, algunas son del campo de la optimización global (Battiti y Tecchiolli, 1995; Tsai et al., 2006; Rocha et al., 2003; Chelouah y Siarry, 2000; Zheng et al., 2005), además de técnicas metaheurísticas basadas en población (Lamos-Sweeney, 2012). Y es Morse y Stanley (2016) quién retoma el uso de métodos metaheurísticos, esto sugiere que otros métodos, como el *Simulated annealing* (SA), pueden tener un buen desempeño

estudiando problemas de minimización, de este modo se plantea que el SA tiene un desempeño computacional competitivo frente al SDG al resolver instancias de un problema de regresión típica de la literatura

1.2 DESCRIPCIÓN DEL PROBLEMA

El problema que se aborda en el presente trabajo presenta diversos estudios en la literatura, a pesar de esto siguen siendo un desafío computacional. Entre los últimos estudios relacionados existen algunos que utilizan programación genética, mostrando las bondades de las metaheurísticas al aplicarlas sobre las NN. A pesar de esto, no existen nuevos estudios que utilizen otras técnicas metaheurísticas para dar solución al problema que presenta la actualización de los pesos en el entrenamiento de las NN.

Para el desarrollo de este estudio se analiza la convergencia de una NN al entrenarla utilizando un método metaheurístico para minimizar el error. Para el desarrollo del estudio es preciso establecer si la convergencia ofrecida por el método ofrece un mejor desempeño respecto a métodos tradicionales y si es una alternativa que se deba considerar.

1.3 SOLUCIÓN PROPUESTA

1.3.1 Características de la solución

El método de aprendizaje basado en simmulated annealing permite la actualización de los pesos de la red. El método supone una alternativa a los métodos tradicionales de aprendizaje para la convergencia de los métodos debido a la independencia que otorga a la actualización de los pesos de las distintas capas y la capacidad de salir de los mínimos locales.

1.3.2 Propósito de la solución

El propósito de la solución es aportar en el campo de las redes neuronales y la clasificación de datos, proporcionando un análisis comparativo de la convergencia de un método metaheuristico aplicado sobre las NN frente a métodos tradicionales.

1.4 OBJETIVOS Y ALCANCES DEL PROYECTO

En ésta sección se presenta el objetivo general, los objetivos específicos además del alcance y limitaciones de la presenta investigación.

1.4.1 Objetivo general

Evaluar el desempeño del algoritmo *simulated annealing* y su efecto sobre el entrenamiento de redes neuronales profundas en comparación con otros métodos. Los métodos que se comparan son: SDG, RMSProp y SA.

1.4.2 Objetivos específicos

Los objetivos establecidos para el presente trabajo son descritos a continuación

1. Definir las reglas de aprendizaje a comparar.
2. Construir los conjuntos de datos de entrada y salida a analizar.
3. Establecer los parámetros de las redes neuronales para la experimentación.

4. Establecer los algoritmos de aprendizaje a comparar.
5. Entrenar las redes con los distintos conjuntos de datos.
6. Establecer las conclusiones del trabajo.

1.4.3 Alcances

1. La arquitectura de las distitnas redes será la misma, variará solo el método de aprendizaje.
2. Los conjunto de datos para el entrenamiento a utilizar son los propuestos en (Morse y Stanley, 2016).

1.5 METODOLOGÍA Y HERRAMIENTAS UTILIZADAS

A continuación se especifica la metodología, herramientas y ambiente de desarrollo que se utilizará para llevar a cabo el presente trabajo.

1.5.1 Metodología de trabajo

Considerando el aspecto investigativo del trabajo, se considera la utilización del método científico. Entre las actividades que componen la metodología, Sampieri (2006) describe los siguientes pasos para desarrollar una investigación:

- Formulación de la hipótesis: Las redes neuronales que adolecen del desvanecimiento del gradiente tienen mejor convergencia al utilizar el algoritmo *simulated annealing* como método de aprendizaje.

- Marco teórico: Una revisión de la literatura donde se aborda el problema planteado, para situarse en el contexto actual de los problemas. Se describirán los principales tópicos que permiten entender el problema y algoritmos de aprendizaje que buscan solucionar el mismo problema.
- Diseño de la solución: Se deberá diseñar el experimento para generar los datos que permitan sustentar las comparaciones entre las distintas redes. Típicamente, los conjuntos de datos que se utilizan son conjuntos utilizados previamente en otras investigaciones para el mismo fin.
- Análisis y verificación de los resultados: Los resultados se analizarán considerando la convergencia de los distintos métodos durante la etapa de entrenamiento de la red.
- Presentación de los resultados: Se presentarán la información que describa los resultados obtenidos y que se consideren pertinentes.
- Conclusiones obtenidas en el desarrollo de la investigación.

1.5.2 Herramientas de desarrollo

Para el desarrollo y ejecución de los experimentos se utilizará un equipo con las siguientes características

Tabla 1.1: Especificaciones del equipo

Sistema Operativo	Solus 2017.04.18.0 64-bit
Procesador	Intel® Core™i5-2450M CPU @ 2.50GHz x 4
RAM	7.7Gb
Gráficos	Intel® Sandybridge Mobile
Almacenamiento	935.6 GB

El software que se utilizará es:

- Lenguaje de programación: Python.
- Sistema de redes neuronales: Keras API (Chollet, 2015).

- Herramienta ofimática: LaTeX.

CAPÍTULO 2. ASPECTOS TEÓRICOS Y REVISIÓN DE LA LITERATURA

En esta sección se abarcan los aspectos relacionados al conocimiento general para la comprensión del presente trabajo (aspectos teóricos) y la revisión de la literatura asociada al trabajo presentado en esta tesis.

2.1 ASPECTOS TEÓRICOS

2.1.1 Redes neuronales multicapas

Una de las características que diferencia a las neuronas biológicas del resto de las células vivas, es su capacidad de comunicación. En la figura 2.1 se puede apreciar un esquema general de una neurona biológica. Las dendritas y el soma (cuerpo celular) reciben las señales de entrada; el cuerpo celular las combina e integra y emite una señal de salida. El axón transporta esas señales a los terminales axónicos, que se encargan de distribuir información a un nuevo conjunto de neuronas. Por lo general, una neurona recibe información de miles de otras neuronas, y a su vez, envía información a otras neuronas, formando una red de conexiones.

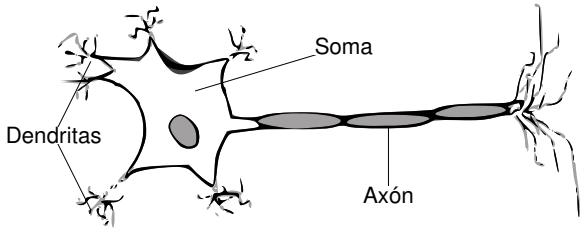

Figura 2.1: Esquema de una neurona biológica.

Una NN es un sistema de procesamiento basado en las conexiones que componen las redes neuronales biológicas y en la forma en que estas se comunican. Cada neurona tiene un estado interno, llamado de *activación*, que es una función de las entradas recibidas y envía su activación

como señal a varias otras neuronas. Cada una está conectada a otras neuronas por medio de enlaces de comunicación dirigidos hacia adelante, formando capas, donde cada conexión tiene un peso asociado como se muestra en la figura 2.2. Los pesos representan la información que está siendo utilizada por la red para resolver un problema.

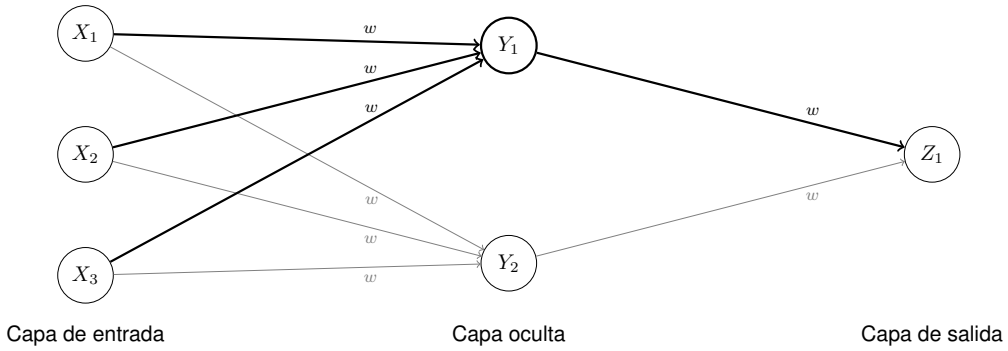

Figura 2.2: Esquema de una red neuronal

La activación de una neurona Y está dado por su función de activación y las entradas. Alguna de las funciones de activación mas comunes se pueden ver en la tabla 2.1. La entrada a la neurona Y corresponde a la suma ponderada de los pesos de las conexiones que llegan hacia Y por la salida de las neuronas de la capa anterior. En la figura 2.2 se puede ver un modelo de tres capas, con una capa oculta. donde las salidas de las neuronas de la capa de entrada llegan hacia las neuronas Y_1 e Y_2 de la capa oculta para aplicar su función de activación respectiva, y sus salida son enviadas a la capa de salida.

Función	**Fórmula**	**Rango**
Identidad	$f(x) = x$	$[-\infty, \infty]$
Lineal por tramos	$f(x) = \begin{cases} -1 & x < -1 \\ a * x & -1 \leq x \leq 1 \\ 1 & x > 1 \end{cases}$	$[-1, 1]$
Sinusoidal	$f(x) = \sin(\omega x + \varphi)$	$[-1, 1]$
Sigmoidal	$f(x) = \frac{1}{1+\exp -x}$	$[0, 1]$
Tangente hiperbólica	$\frac{1-\exp(-x)}{1+\exp(-x)}$	$[-1, 1]$

Tabla 2.1: Algunas funciones de activaciones.

Las NN multicapa definen una relación entre la entrada y la salida. Esta relación se obtiene propagando hacia adelante los valores de las variables de entrada, es por esto que también se

les llama redes *feedforward*. Cada neurona de la red procesa la entrada recibida y produce una respuesta que se propaga, mediante las conexiones, hacia las neuronas de la capa siguiente.

Existen dos fases importante dentro del modelo

- Fase de entrenamiento: Se usa un conjunto de datos o patrones de entrenamiento para determinar los pesos que definen el modelo de la NN. Se calculan de manera iterativa, de acuerdo con los valores de entrenamiento, con el objeto de minimizar el error cometido entre la salida obtenida por la NN y la salida deseada.

  Los pesos óptimos se obtienen minimizando una función. Uno de los criterios utilizados es la minimización del error cuadrático medio entre el valor de salida y el valor real esperado.

- Fase de prueba: Durante el entrenamiento, el modelo se ajusta al conjunto de entrenamiento, perdiendo la habilidad de generalizar su aprendizaje a casos nuevos, a esta situación se le llama sobreajuste.

  Para evitar el sobreajuste, se utiliza un segundo grupo de datos diferentes, el conjunto de validación, que permitirá controlar el proceso de aprendizaje.

2.1.2 El algoritmo de entrenamiento por retropropagación y el desvanecimiento del gradiente

El algoritmo de retropropagación del error, también conocido como la regla delta, fue el primer algoritmo de entrenamiento para redes multicapas (Werbos, 1974; Rumelhart et al., 1986a). El término retropropagación es utilizado debido a la forma de implementar el método del gradiente en las redes multicapa, pues el error cometido en la salida de la red es propagado hacia atrás, transformándolo en un error para cada una de las neuronas ocultas de la red. El entrenamiento de una red por retropropagación implica tres etapas: la propagación del patrón de entrada, el cálculo del error y su propagación hacia las capas anteriores, y el ajuste de los pesos. Después del entrenamiento, la aplicación de la red implica solamente los cálculos de la fase de propagación. En caso de que el entrenamiento sea lento, una red ya entrenada puede producir su salida rápidamente.

El funcionamiento del agoritmo de retropropagación se puede apreciar en la figura 2.3. Se

representa en azul primera fase del algoritmo, donde la entrada de la red se propaga hacia la salida a través de las neuronas transformandola en cada neurona de la red. La segunda fase, en verde, muestra como se propaga, desde la salida, el error hacia las capas anteriores. Finalmente, se puede ver en rojo como el algoritmo actualiza los pesos de la red utilizando el error que genera la red.

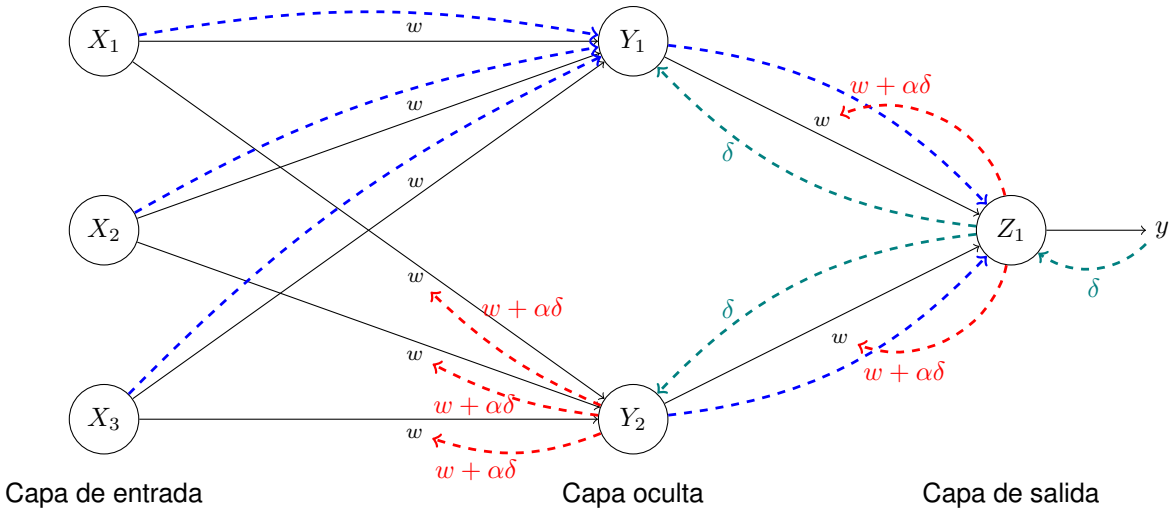

Figura 2.3: Esquema del algoritmo de retropropagación.

Si un perceptrón multicapa con C capas y n_c neuronas en la capa c, donde $W_c = (w_{ij}^c)$ es la matriz de pesos, w_{ij}^c representará el peso de la conexion de la neurona i de la capa c hasta la neurona j de la capa siguiente. Denotaremos a_i^c a la activación de la neurona i de la capa c que se calcula de la siguiente manera:

- **Activación de una neurona de la capa de entrada**: Las neuronas se encargan de transmitir la entrada recibida, por lo tanto

$$a_i^1 = x_i, i = 1, 2, \cdots, n$$

  donde $X = (x_1, x_2, \cdots, x_n)$ representa el vector de entrada.

- **Activación de una neurona de la capa oculta**: Las neuronas de una capa oculta procesa la información recibida aplicando la función f a la suma de los productos de la entrada por sus pesos, es decir

$$a_i^c = f\left(\sum_{j=1}^{n_{c-1}} w_{ji}^{c-1} a_j^{c-1} + \theta_i^c\right), i = 1, 2, \cdots, n_c; c = 2, 3, \cdots, C-1$$

  donde a_j^{c-1} es la salida de la capa anterior a c.

- **Activación de una neurona de la capa de salida**: La activación de una neurona de la capa de salida viene dada por la función f aplicada a la suma de los productos de la entrada por sus pesos, es decir

$$y_i = a_i^c = f\left(\sum_{j=1}^{n_{c-1}} w_{ji}^{C-1} a_j^{C-1} + \theta_i^C\right), i = 1, \cdots, n_c$$

donde $Y = (y_1, y_2, \cdots, y_{n_c})$ es el vector de salida.

La función f es la función de activación de la neurona. Aunque existe gran variedad de funciones de activación (ver tabla 2.1), las funciones de activación mas utilizadas son la sigmoidal y la tangente hiperbólica, descritas en las escuaciones 2.1 y 2.2 respectivamente.

$$f_{sigm}(x) = \frac{1}{1+\exp(-x)} \tag{2.1}$$

$$f_{tanh}(x) = \frac{1-\exp(-x)}{1+\exp(-x)} \tag{2.2}$$

Ambas funciones poseen como imagen un intervalo de valores entre $[0, 1]$ y $[-1, 1]$ como se observa en la figura 2.4.

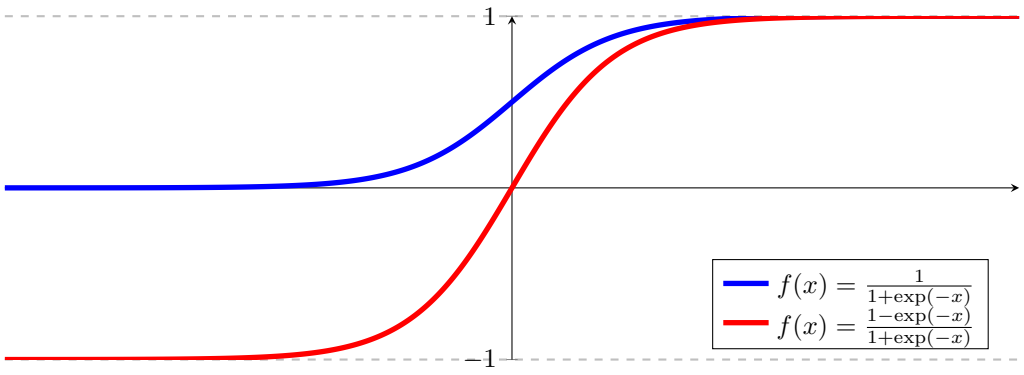

Figura 2.4: Funciones de activación mas utilizadas.

Las NN multicapa actualizan sus pesos en función de una regla de aprendizaje, de tal manera que los nuevos pesos permitan reducir el error de salida. Por tanto, para cada patrón de entrada necesario disponer de una de salida deseada. El objetivo es que la salida de la red sea lo más próxima posible a la salida deseada, debido a esto es que el aprendizaje de la red se describe como un problema de minimización de la siguiente manera

$$\min_W E$$

donde W es el conjunto de parámetros de la red (pesos y umbrales) y E es una función de error que evalúa la diferencia entre las salidas de la red y las salidas deseadas. En la mayor parte de los casos, la función de error se define como:

$$E = \frac{1}{N}\sum_{i=1}^{N} e(i) \tag{2.3}$$

Donde N es el número de muestras y $e(n)$ es el error cometido por la red para el patrón i, definido de la siguiente manera

$$e(i) = \frac{1}{n_C}\sum_{j=1}^{n_C}(s_j(i) - y^j(n))^2 \tag{2.4}$$

Siendo $Y(i) = (y_1(i), y_2(i), \cdots, y_{n_C}(i))$ y $S(i) = (s_1(i), s_2(i), \cdots, s_{n_C}(i))$ los vectores de salida y salidas deseadas para el patrón i respectivamente.

De esta manera, si W^* es un mínimo de la función de error E, en dicho punto el error será cercano a cero, y en consecuencia, la salida de la red será próxima a la salida deseada. La presencia de funciones de activación no lineales hace que la respuesta de la red sea no lineal respecto a los parámetros ajustables, por lo que el problema de minimización es un problema no lineal y se hace necesario el uso de técnicas de optimización no lineales para su resolución.

Las técnicas de actualización utilizadas suelen basarse en la actualización de los parámetros de la red mediante la determinación de una dirección de búsqueda. En el caso de las NN multicapa, la dirección de búsqueda más utilizada se basa en la dirección contraria del gradiente de la función de error E, el método de gradiente descendente. El procedimiento está basado en una sucesión de minimizaciones del error $e(i)$ por cada patrón, en lugar de minimizar el error total E de la red. Aplicando el método cada parámetro w se modifica según la siguiente regla de aprendizaje

$$w(i) = w(i-1) - \alpha\frac{\partial e(i)}{\partial w} \tag{2.5}$$

donde $e(i)$ es el error para el patrón de entrada i dado por la ecuación 2.4, y α es la tasa de aprendizaje, éste último determina el desplazamiento en la superficie del error.

A medida que el error se propaga a través de la red, los pesos se actualizarán según la ecuación 2.5, utilizando funciones cuyo gradiente se encuentra entre 0 y 1. Debido a que estos gradientes se multiplican durante la retropropagación, tienden a *desvanecerse* a través de las capas, y en una NN profunda esto evita que los pesos de las primeras capas ocultas se actualicen minimamente.

Si se tiene una NN, la activación de una neurona de una capa intermedia i con función de

activación f es $y^i(t) = f_i(net_i(t))$ donde

$$net_i(t) = \sum_j w_{ji} y^j(t-1)$$

es la entrada a la neurona. Además w_{ji} es el peso de la conexión desde la unidad j de la capa anterior hasta la unidad i de la capa actual, $d_k(t)$ será la respuesta esperada de la unidad k de la capa de salida en el tiempo t. Usando el error cuadrático medio (*Mean square error*, MSE), el error de k será

$$E_k(t) = (d_k(t) - y^k(t))^2$$

En un tiempo $\tau \leq t$ cualquiera, el error de una neurona j que no sea una neurona de entrada es la suma de los errores externos y el error propagado hacia atrás desde la neurona previa será

$$\vartheta_j(\tau) = f_j'(net_j(\tau)) \left(E_j(\tau) + \sum_i w_{ij} \vartheta_i(\tau + 1) \right)$$

El peso actualizado en el tiempo τ resulta $w_{jl}^{new} = w_{jl}^{old} + \alpha \vartheta_j(\tau) y^l(\tau - 1)$ donde α es la tasa de aprendizaje, y l es una unidad arbitraria conectada a la unidad j.

La propagación hacia atrás de un error que ocurre en una unidad u en un tiempo t hacia una unidad v para q pasos, escala el error de la siguiente manera

$$\frac{\partial \vartheta_v(t-q)}{\partial \vartheta_u(t)} = \begin{cases} f_v'(net_v(t-1)) w_{uv} & q = 1 \\ f_v'(net_v(t-q)) \sum_{l=1}^{n} \frac{\partial \vartheta(t-q+1)}{\partial \vartheta_u(t)} w_{lv} & q > 1 \end{cases} \tag{2.6}$$

Con $l_q = v$ y $l_0 = u$, el factor de escalamiento es

$$\frac{\partial \vartheta_v(t-q)}{\partial \vartheta_u(t)} = \sum_{l_1=1}^{n} \cdots \sum_{l_{q-1}=1}^{n} \prod_{m=1}^{q} f_{l_m}'(net_{l_m}(t-m)) w_{l_m l_{m-1}} \tag{2.7}$$

La sumatoria de los n^{q-1} términos $\prod_{m=1}^{q} f_{l_m}'(net_{l_m}(t-m)) w_{l_m l_{m-1}}$ escalan el error. Los distintos términos pueden tener signos diferentes, por lo tanto, el aumento del número de unidades n no implica un incremento del error absoluto. Pero con mas unidades se incrementa la expectativa de que el valor absoluto del error aumente. Si $\rho(m, l_m, l_{m-1}) := |f_{l_m}'(net_{l_m}(t-m)) w_{l_m l_{m-1}}| < 1.0$ para todo m, el producto en (2.7) decrece exponencialmente con q, es decir, el error se desvanece como muestra la figura 2.5. Un error que se desvanece a lo largo del flujo casi no tiene efecto en la actualización de los pesos.

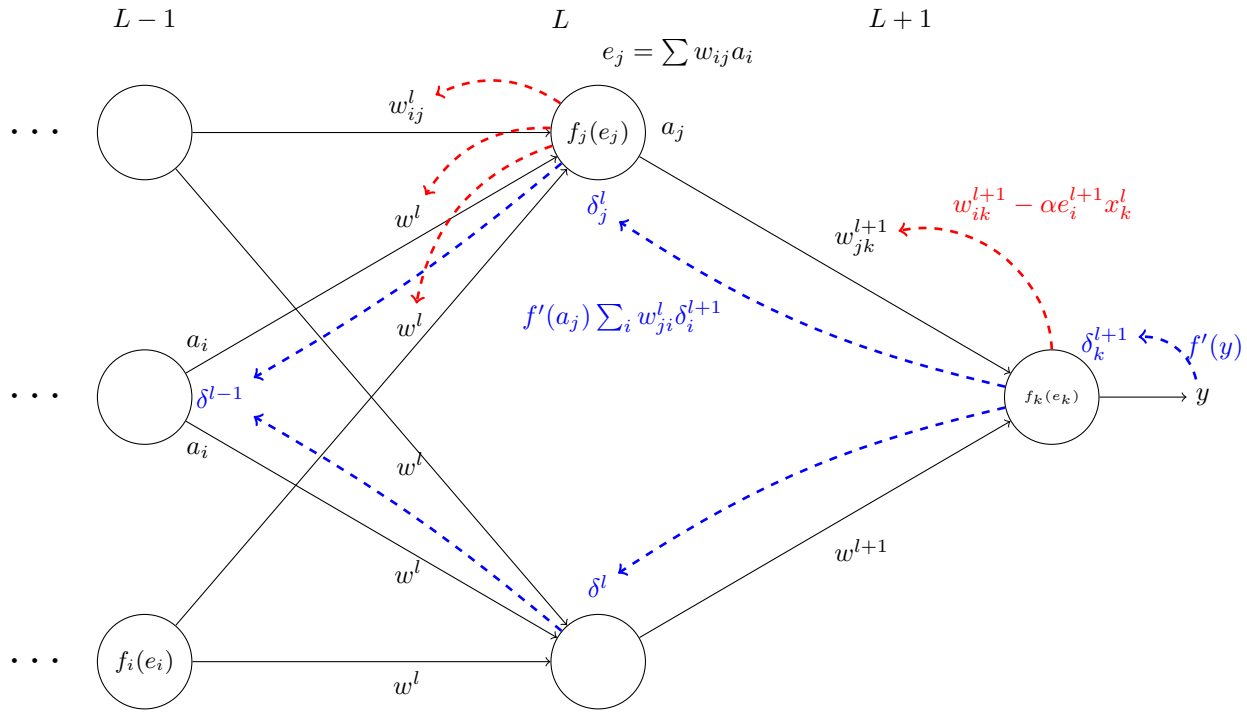

Figura 2.5: Gradiente descendente

2.2 REVISIÓN DE LA LITERATURA

2.2.1 El gradiente estocástico descendente

El gradiente estocástico descendente es una drástica simplificación del método tradicional. En vez de calcular el gradiente exacto de la función de error E, en cada iteración se estima el valor del gradiente basado en un ejemplo escogido de manera aleatoria. El algoritmo de descenso de gradiente tradicional actualiza los parámetros W según la siguiente expresión

$$w^{(i)} = w^{(i-1)} - \alpha \frac{\partial E^{(i)}}{\partial w}$$

y en el algoritmo 1 se describe su funcionamiento en términos generales.

Algoritmo 1: Algoritmo del gradiente descendente

1 **para** *uno o más epocas* **hacer**
2 **para** *cada peso* j **hacer**
3 $w_j := w + \Delta w_j$ donde $\Delta w_j = \alpha \sum_i (target^i - output^i) x_j^i$

Donde el resultado obtenido es aproximado evaluando el costo y el gradiente sobre el conjunto de entrenamiento completo. El gradiente estocástico descendente (SGD) elimina la necesidad de analizar el conjunto de entrenamiento en cada iteración, calculando el gradiente de los parámetros utilizando sólo un único, o pocos, ejemplos de entrenamiento como muestra el algoritmo 2. La forma de actualización que propone el método SGD está dada por la ecuación 2.8.

$$w^{(i)} = w^{(i-1)} - \alpha \frac{\partial E(x^{(i)}, y^{(i)})}{\partial w} \tag{2.8}$$

Con un par $(x^{(i)}, y^{(i)})$ del conjunto de entrenamiento.

Generalmente cada actualización de parámetros en SGD analiza algunos ejemplos de entrenamiento en lugar de un solo ejemplo. Esto reduce la varianza en la actualización de parámetros y puede conducir a una convergencia más estable. Un tamaño de minibatch típico es 256, aunque el tamaño óptimo del minibatch puede variar para diferentes aplicaciones y arquitecturas.

Un punto final pero importante con respecto a SGD es el orden en que presentamos los datos al algoritmo. Si los datos se dan en algún orden significativo, esto puede sesgar el gradiente y conducir a la convergencia pobres. Generalmente un buen método para evitar esto es mezclar aleatoriamente los datos antes de cada época de entrenamiento.

Algoritmo 2: Algoritmo del gradiente descendente estocástico

1 **para** *uno o más epocas o el costo mínimo deseado ha sido alcanzado* **hacer**
2 **para** *ejemplo de entrenamiento* i *aleatorio* **hacer**
3 **para** *cada peso* j **hacer**
4 $w_j = w + \alpha(target^i - output^i) x_j^i$;

2.2.2 Retropropagación resiliente y RMSProp

En las capas ocultas de las NN se utilizan con frecuencia funciones de activación sigmoidales, y estas reducen la entrada a un rango finito de salida. Se caracterizan por sus pendientes próximas a cero para entradas muy grandes, y esto representa un problema cuando se utiliza el gradiente descendente para el entrenamiento, pues se acentúa el problema del desvanecimiento del gradiente.

Riedmiller y Braun (1993) presenta el algoritmo de retropropagación resiliente (*Resilient retropropagation*, RPROP), con el que busca eliminar los efectos del desvanecimiento del gradiente, esto mediante el uso del signo de la derivada para determinar la dirección del ajuste de los pesos, haciendo que la magnitud de la derivada no tenga efecto sobre la actualización de los pesos de la NN. Los pesos de la NN se incrementarán por un factor Δi cuando la derivada de la función respecto a dicho peso tenga el mismo signo que las dos iteraciones anteriores. Mientras que si la derivada de la función respecto a dicho peso cambia de signo respecto de la iteración anterior los pesos se decrementarán por un factor Δ_d. En caso de que la derivada de la función respecto de dicho peso sea igual a cero, el valor de actualización no varía.

En 2012 Tieleman y Hinton (2012) presentan una variante del algoritmo RPROP, el algoritmo de la raíz cuadratica media de retropropagación resiliente (*Root mean square resilient retropropagation*, RMSPROP). Se propone mantener una media móvil del cuadrado del gradiente para cada peso

$$MeanSquare(w,t) \quad = \quad 0.9MeanSquare(w,t-1) + 0.1\frac{\partial E}{\partial w^{(t)}}^2 \tag{2.9}$$

Y dividiendo el gradiente por $\sqrt{MeanSquare(w,t)}$ hace que el aprendizaje funcione de mejor manera.

2.2.3 Heurísticas y meta-heurísticas

Hoy en día, resolver problemas computacionalmente complejos precisa el desarrollo de algoritmos más avanzados. Los algoritmos exactos a menudo utilizan una gran cantidad de tiempo debido al tamaño del espacio de soluciones factibles. Para resolver este inconveniente, se han diseñado algoritmos aproximados basados en el uso de heurísticas y meta-heurísticas, los que permiten

encontrar soluciones que se aproximan a la mejor solución. Estos algoritmos utilizan funciones que están diseñadas para encontrar el espacio de soluciones de forma inteligente.

La figura 2.6 muestra la clasificación de diferentes problemas de optimización, los que se categorizan en: algoritmos exactos y algoritmos aproximados (Desale et al., 2015)

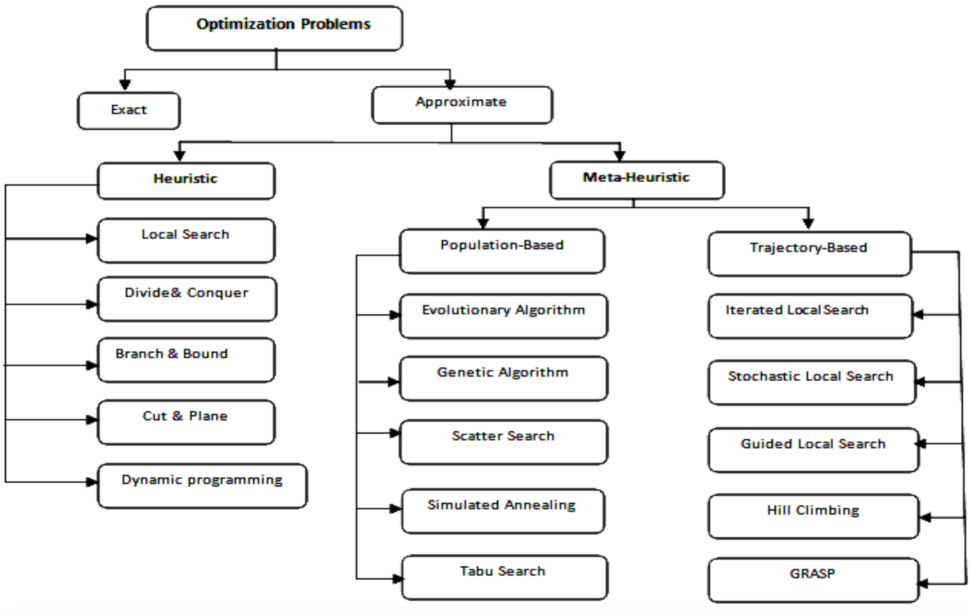

Figura 2.6: Métodos para resolver problemas de optimización (Desale et al., 2015).

Los algoritmos meta-heurísticos son un proceso iterativo que guía una heurística para explorar el espacio de búsqueda. Estos tipos de algoritmos son utilizados para encontrar una solución en un espacio de búsqueda discreto en problemas de optimización combinatoria.

Simulated Annealing

El método *simulated annealing* (SA) es un algoritmo de búsqueda meta-heurística para problemas de optimización combinatoria introducido por (Kirkpatrick et al., 1983) y (Černý, 1985). La técnica está basada en el proceso de calentamiento de un metal, vidrio o cristal hasta su punto de fusión, para luego enfriarlo hasta obtener una estructura cristalina. El algoritmo 3 decribe el procedimiento de la siguiente manera: dada una solución inicial se genera un conjunto de

soluciones que son una perturbación de la solución actual, en el caso de que al evaluar la perturbación se obtenga un mejor desempleo que la solución actual, se actualizará la solución actual; y en caso contrario, existirá una probabilidad de que la solución actual sea actualizada de igual manera.

Algoritmo 3: Simulated annealing

Datos: Temperatura T, constante de Boltzmann k, factor de reducción c

```
1  Seleccionar un vector solución x_0
2  mientras El criterio no se cumpla hacer
3      mientras Existan soluciones en el conjunto hacer
4          Seleccionar una solución x_0 + Δx
5          si f(x_0 + Δx) < f(x_0) entonces
6              f_new = f(x_0 + Δx); x_0 = x_0 + Δx
7          en otro caso
8              Δf = f(x_0 + Δx) − f(x_0)
9              si rand(0, 1) > exp(−Δf/kT) entonces
10                 f_new = f(x_0 + Δx); x_0 = x_0 + Δx
11             en otro caso
12                 f_new = f(x_0)
13             fin
14         fin
15         f = f_new; T = cT
16     fin
17 fin
```

REFERENCIAS BIBLIOGRÁFICAS

Adam-Bourdarios, C., Cowan, G., Germain, C., Guyon, I., Kégl, B., y Rousseau, D. (2015, 13 Dec). The Higgs boson machine learning challenge. En G. Cowan, C. Germain, I. Guyon, B. Kégl, y D. Rousseau (Eds.), *Proceedings of the nips 2014 workshop on high-energy physics and machine learning* (Vol. 42, pp. 19–55). Montreal, Canada: PMLR. Descargado de http://proceedings.mlr.press/v42/cowa14.html

Al-Baali, M. (1998). Numerical experience with a class of self-scaling quasi-newton algorithms. *Journal of Optimization Theory and Applications*, *96*(3), 533–553. doi: 10.1023/A:1022608410710

Arel, I., Rose, D. C., y Karnowski, T. P. (2010, nov). Research frontier: Deep machine learning–a new frontier in artificial intelligence research. *Comp. Intell. Mag.*, *5*(4), 13–18. Descargado de http://dx.doi.org/10.1109/MCI.2010.938364 doi: 10.1109/MCI.2010.938364

Barzilai, J., y Borwein, J. M. (1988). Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, *8*(1), 141. doi: 10.1093/imanum/8.1.141

Battiti, R., y Tecchiolli, G. (1995, Sep). Training neural nets with the reactive tabu search. *IEEE Transactions on Neural Networks*, *6*(5), 1185-1200. doi: 10.1109/72.410361

Bengio, Y. (2009, enero). Learning deep architectures for ai. *Found. Trends Mach. Learn.*, *2*(1), 1–127. Descargado de http://dx.doi.org/10.1561/2200000006 doi: 10.1561/2200000006

Bengio, Y., Courville, A., y Vincent, P. (2013, Aug). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1798-1828. doi: 10.1109/TPAMI.2013.50

Bengio, Y., Lamblin, P., Popovici, D., y Larochelle, H. (2006). Greedy layer-wise training of deep networks. En *Proceedings of the 19th international conference on neural information processing systems* (pp. 153–160). Cambridge, MA, USA: MIT Press. Descargado de http://dl.acm.org/citation.cfm?id=2976456.2976476

Bengio, Y., y LeCun, Y. (2007). Scaling learning algorithms towards AI. En L. Bottou, O. Chapelle, D. DeCoste, y J. Weston (Eds.), *Large-scale kernel machines*. MIT Press. Descargado de http://yann.lecun.com/exdb/publis/pdf/bengio-lecun-07.pdf

Bianchini, M., y Gori, M. (1996). Optimal learning in artificial neural networks: A review of theoretical results. *Neurocomputing*, *13*(2–4), 313 - 346. Descargado de http://www.sciencedirect.com/science/article/pii/0925231295000321 (Soft Computing) doi: https://doi.org/10.1016/0925-2312(95)00032-1

Birgin, E. G., y Martínez, J. M. (2001). A spectral conjugate gradient method for unconstrained optimization. *Applied Mathematics and Optimization*, *43*(2), 117–128. Descargado de http://dx.doi.org/10.1007/s00245-001-0003-0 doi: 10.1007/s00245-001-0003-0

Bishop, C. M. (1995). *Neural networks for pattern recognition*. New York, NY, USA: Oxford University Press, Inc.

Bordes, A., Chopra, S., y Weston, J. (2014). Question answering with subgraph embeddings. *CoRR*, *abs/1406.3676*. Descargado de http://arxiv.org/abs/1406.3676

Cauchy, A.-L. (1847, 18 de octubre). Méthode générale pour la résolution des systèmes d'équations simultanées. *Compte Rendu des S'eances de L'Acad'emie des Sciences XXV*, *S'erie A*(25), 536–538.

Černý, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, *45*(1), 41–51. Descargado de http://dx.doi.org/10.1007/BF00940812 doi: 10.1007/BF00940812

Charalambous, C. (1992, June). Conjugate gradient algorithm for efficient training of artificial neural networks. *IEE Proceedings G - Circuits, Devices and Systems*, *139*(3), 301-310. doi: 10.1049/ip-g-2.1992.0050

Chelouah, R., y Siarry, P. (2000). Tabu search applied to global optimization. *European Journal of Operational Research*, *123*(2), 256 - 270. Descargado de http://www.sciencedirect.com/science/article/pii/S0377221799002556 doi: http://dx.doi.org/10.1016/S0377-2217(99)00255-6

Chollet, F. (2015). *Keras.* https://github.com/fchollet/keras. GitHub.

Ciodaro, T., Deva, D., de Seixas, J. M., y Damazio, D. (2012). Online particle detection with neural networks based on topological calorimetry information. *Journal of Physics: Conference Series*, *368*(1), 012030. Descargado de http://stacks.iop.org/1742-6596/368/i=1/a=012030

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., y Kuksa, P. (2011, noviembre). Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, *12*, 2493–2537. Descargado de http://dl.acm.org/citation.cfm?id=1953048.2078186

Desale, S., Rasool, A., Andhale, S., y Rane, P. (2015, 5). Heuristic and meta-heuristic algorithms and their relevance to the real world: A survey. *International journal of computer engineering in research trends*, *2*, 296-304.

Farabet, C., Couprie, C., Najman, L., y LeCun, Y. (2013, Aug). Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1915-1929. doi: 10.1109/TPAMI.2012.231

Fletcher, R., y Reeves, C. M. (1964, 1 de febrero). Function minimization by conjugate gradients. *The Computer Journal*, *7*(2), 149–154. Descargado de `http://dx.doi.org/10.1093/comjnl/7.2.149` doi: 10.1093/comjnl/7.2.149

Gori, M., y Tesi, A. (1992, Jan). On the problem of local minima in backpropagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *14*(1), 76-86. doi: 10.1109/34.107014

Gorin, A., y Mammone, R. J. (1994, Jan). Introduction to the special issue on neural networks for speech processing. *IEEE Transactions on Speech and Audio Processing*, *2*(1), 113-114. doi: 10.1109/89.260355

Grippo, L. (1994). A class of unconstrained minimization methods for neural network training. *Optimization Methods and Software*, *4*(2), 135-150. doi: 10.1080/10556789408805583

Helmstaedter, M., Briggman, K. L., Turaga, S. C., Jain, V., Seung, H. S., y Denk, W. (2013, aug). Connectomic reconstruction of the inner plexiform layer in the mouse retina. *Nature*, *500*(7461), 168–174. Descargado de `https://doi.org/10.1038/nature12346` doi: 10.1038/nature12346

Hestenes, M. R., y Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, *49*, 409–436.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., r. Mohamed, A., Jaitly, N., ... Kingsbury, B. (2012, Nov). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, *29*(6), 82-97. doi: 10.1109/MSP.2012.2205597

Hinton, G. E., Osindero, S., y Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*(7), 1527-1554. Descargado de `http://dx.doi.org/10.1162/neco.2006.18.7.1527` (PMID: 16764513) doi: 10.1162/neco.2006.18.7.1527

Hinton, G. E., y Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507. Descargado de http://science.sciencemag.org/content/313/5786/504 doi: 10.1126/science.1127647

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, *4*(2), 251 - 257. Descargado de http://www.sciencedirect.com/science/article/pii/089360809190009T doi: https://doi.org/10.1016/0893-6080(91)90009-T

Huang, H. Y. (1970). Unified approach to quadratically convergent algorithms for function minimization. *Journal of Optimization Theory and Applications*, *5*(6), 405–423. Descargado de http://dx.doi.org/10.1007/BF00927440 doi: 10.1007/BF00927440

Hwang, J.-N., Kung, S.-Y., Niranjan, M., y Principe, J. C. (1997, Nov). The past, present, and future of neural networks for signal processing. *IEEE Signal Processing Magazine*, *14*(6), 28-48. doi: 10.1109/79.637299

Jacobs, R. A. (1988). Increased rates of convergence through learning rate adaptation. *Neural Networks*, *1*(4), 295 - 307. doi: https://doi.org/10.1016/0893-6080(88)90003-2

Jain, A. K., Duin, R. P. W., y Mao, J. (2000, enero). Statistical pattern recognition: A review. *IEEE Trans. Pattern Anal. Mach. Intell.*, *22*(1), 4–37. Descargado de http://dx.doi.org/10.1109/34.824819 doi: 10.1109/34.824819

Jean, S., Cho, K., Memisevic, R., y Bengio, Y. (2014). On using very large target vocabulary for neural machine translation. *CoRR*, *abs/1412.2007*. Descargado de http://arxiv.org/abs/1412.2007

Kirkpatrick, S., Gelatt, C. D., y Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671–680. Descargado de http://science.sciencemag.org/content/220/4598/671 doi: 10.1126/science.220.4598.671

Krizhevsky, A., Sutskever, I., y Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. En *Proceedings of the 25th international conference on neural nformation processing systems* (pp. 1097–1105). USA: Curran Associates Inc. Descargado de http://dl.acm.org/citation.cfm?id=2999134.2999257

Lamos-Sweeney, J. (2012). *Deep learning using genetic algorithms*. Descargado de https://books.google.cl/books?id=RGpPMwEACAAJ

Le, Q. V., Ranzato, M., Monga, R., Devin, M., Chen, K., Corrado, G. S., ... Ng, A. Y. (2012). Building high-level features using large scale unsupervised learning. En *Proceedings of the 29th*

international coference on international conference on machine learning (pp. 507–514). USA: Omnipress. Descargado de http://dl.acm.org/citation.cfm?id=3042573.3042641

LeCun, Y., Bengio, Y., y Hinton, G. (2015, may). Deep learning. *Nature*, *521*(7553), 436–444. Descargado de https://doi.org/10.1038/nature14539 doi: 10.1038/nature14539

Leung, M. K. K., Xiong, H. Y., Lee, L. J., y Frey, B. J. (2014, jun). Deep learning of the tissue-regulated splicing code. *Bioinformatics*, *30*(12), i121–i129. Descargado de https://doi.org/10.1093/bioinformatics/btu277 doi: 10.1093/bioinformatics/btu277

Ma, J., Sheridan, R. P., Liaw, A., Dahl, G. E., y Svetnik, V. (2015). Deep neural nets as a method for quantitative structure–activity relationships. *Journal of Chemical Information and Modeling*, *55*(2), 263-274. Descargado de http://dx.doi.org/10.1021/ci500747n (PMID: 25635324) doi: 10.1021/ci500747n

Magoulas, G. D., Vrahatis, M. N., y Androulakis, G. S. (1997). Effective backpropagation training with variable stepsize. *Neural Networks*, *10*(1), 69 - 82. Descargado de http://www.sciencedirect.com/science/article/pii/S0893608096000524 doi: https://doi.org/10.1016/S0893-6080(96)00052-4

Mikolov, T., Deoras, A., Povey, D., Burget, L., y Cernocky, J. H. (2011, December). Strategies for training large scale neural network language models. IEEE Automatic Speech Recognition and Understanding Workshop. Descargado de https://www.microsoft.com/en-us/research/publication/strategies-for-training-large-scale-neural-network-language-models/

Mitra, S., y Hayashi, Y. (2006, Sept). Bioinformatics with soft computing. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *36*(5), 616-635. doi: 10.1109/TSMCC.2006.879384

Morse, G., y Stanley, K. O. (2016). Simple evolutionary optimization can rival stochastic gradient descent in neural networks. En *Proceedings of the genetic and evolutionary computation conference 2016* (pp. 477–484). New York, NY, USA: ACM. Descargado de http://doi.acm.org/10.1145/2908812.2908916 doi: 10.1145/2908812.2908916

Møller, M. F. (1993). A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, *6*(4), 525 - 533. Descargado de http://www.sciencedirect.com/science/article/pii/S0893608005800565 doi: https://doi.org/10.1016/S0893-6080(05)80056-5

Nguyen, D., y Widrow, B. (1990, June). Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. En *1990 ijcnn international joint conference on neural networks* (p. 21-26 vol.3). doi: 10.1109/IJCNN.1990.137819

Nocedal, J., y Wright, S. (2006). *Numerical optimization (springer series in operations research and financial engineering)*. Springer.

Nocedal, J., y Yuan, Y.-x. (1993). Analysis of a self-scaling quasi-newton method. *Mathematical Programming*, *61*(1), 19–37. doi: 10.1007/BF01582136

Oren, S. (1972). *Self-scaling variable metric algorithms for unconstrained minimization*. Department of Engineering-Economic Systems, Stanford University.

Oren, S. S., y Luenberger, D. G. (1974). Self-scaling variable metric (ssvm) algorithms. part i: Criteria and sufficient conditions for scaling a class of algorithms. *Management Science*, *20*(5), 845-862. Descargado de `http://www.jstor.org/stable/2630094`

Peng, C. C., y Magoulas, G. D. (2007, Oct). Adaptive nonmonotone conjugate gradient training algorithm for recurrent neural networks. En *19th ieee international conference on tools with artificial intelligence(ictai 2007)* (Vol. 2, p. 374-381). doi: 10.1109/ICTAI.2007.126

Plagianakos, V., Sotiropoulos, D., y Vrahatis, M. (1998). Automatic adaptation of learning rate for backpropagation neural networks. *Recent Advances in Circuits and Systems*, *337*.

Plagianakos, V. P., Magoulas, G. D., y Vrahatis, M. N. (2002, Nov). Deterministic nonmonotone strategies for effective training of multilayer perceptrons. *IEEE Transactions on Neural Networks*, *13*(6), 1268-1284. doi: 10.1109/TNN.2002.804225

Polak E., R. G. (1969). Note sur la convergence de méthodes de directions conjuguées. *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique*, *3*(R1), 35-43. Descargado de `http://eudml.org/doc/193115`

Ranzato, M., lan Boureau, Y., y Cun, Y. L. (2007). Sparse feature learning for deep belief networks. En J. Platt, D. Koller, Y. Singer, y S. Roweis (Eds.), *Advances in neural information processing systems 20* (pp. 1185–1192). Cambridge, MA: MIT Press. Descargado de `http://books.nips.cc/papers/files/nips20/NIPS2007_1118.pdf`

Riedmiller, M., y Braun, H. (1993). A direct adaptive method for faster backpropagation learning: the rprop algorithm. En *Ieee international conference on neural networks* (p. 586-591 vol.1). doi: 10.1109/ICNN.1993.298623

Rocha, M., Cortez, P., y Neves, J. (2003). Evolutionary neural network learning. En F. M. Pires y S. Abreu (Eds.), *Progress in artificial intelligence: 11th portuguese conference on artificial intelligence, epia 2003, beja, portugal, december 4-7, 2003. proceedings* (pp. 24–28). Berlin,

Heidelberg: Springer Berlin Heidelberg. Descargado de http://dx.doi.org/10.1007/978-3-540-24580-3_10 doi: 10.1007/978-3-540-24580-3_10

Rumelhart, D. E., Hinton, G. E., y Williams, R. J. (1986a). Learning representations by back-propagating errors. *Nature*, *323*, 533–536.

Rumelhart, D. E., Hinton, G. E., y Williams, R. J. (1986b). Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. En D. E. Rumelhart, J. L. McClelland, y C. PDP Research Group (Eds.), (pp. 318–362). Cambridge, MA, USA: MIT Press. Descargado de http://dl.acm.org/citation.cfm?id=104279.104293

Sainath, T. N., r. Mohamed, A., Kingsbury, B., y Ramabhadran, B. (2013, May). Deep convolutional neural networks for lvcsr. En *2013 ieee international conference on acoustics, speech and signal processing* (p. 8614-8618). doi: 10.1109/ICASSP.2013.6639347

Sampieri, R. (2006). *Metodología de la investigación*. México: McGraw Hill.

Selmic, R. R., y Lewis, F. L. (2002, mayo). Neural-network approximation of piecewise continuous functions: Application to friction compensation. *Trans. Neur. Netw.*, *13*(3), 745–751. Descargado de http://dx.doi.org/10.1109/TNN.2002.1000141 doi: 10.1109/TNN.2002.1000141

Sotiropoulos, D., Kostopoulos, A., y Grapsa, T. (2002). A spectral version of perry's conjugate gradient method for neural network training. En *Proceedings of 4th gracm congress on computational mechanics* (Vol. 1, pp. 291–298).

Sutskever, I., Vinyals, O., y Le, Q. V. (2014). Sequence to sequence learning with neural networks. *CoRR*, *abs/1409.3215*. Descargado de http://arxiv.org/abs/1409.3215

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... Rabinovich, A. (2015, June). *Going deeper with convolutions.* doi: 10.1109/CVPR.2015.7298594

Tieleman, T., y Hinton, G. (2012). *Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude.* COURSERA: Neural Networks for Machine Learning.

Tompson, J., Jain, A., LeCun, Y., y Bregler, C. (2014). Joint training of a convolutional network and a graphical model for human pose estimation. *CoRR*, *abs/1406.2984*. Descargado de http://arxiv.org/abs/1406.2984

Tsai, J.-T., Chou, J.-H., y Liu, T.-K. (2006, Jan). Tuning the structure and parameters of a neural network by using hybrid taguchi-genetic algorithm. *IEEE Transactions on Neural Networks*, *17*(1), 69-80. doi: 10.1109/TNN.2005.860885

Vogl, T. P., Mangis, J. K., Rigler, A. K., Zink, W. T., y Alkon, D. L. (1988). Accelerating the convergence of the back-propagation method. *Biological Cybernetics*, *59*(4), 257–263. doi: 10.1007/BF00332914

Werbos, P. J. (1974). *Beyond regression: New tools for prediction and analysis in the behavioral sciences* (Tesis Doctoral no publicada). Harvard University.

Xiong, H. Y., Alipanahi, B., Lee, L. J., Bretschneider, H., Merico, D., Yuen, R. K. C., ... Frey, B. J. (2015). The human splicing code reveals new insights into the genetic determinants of disease. *Science*, *347*(6218). Descargado de `http://science.sciencemag.org/content/347/6218/1254806` doi: 10.1126/science.1254806

Yin, H. X., y Du, D. L. (2007). The global convergence of self-scaling bfgs algorithmwith nonmonotone line search forunconstrained nonconvex optimization problems. *Acta Mathematica Sinica, English Series*, *23*(7), 1233–1240. doi: 10.1007/s10114-005-0837-5

Zhang, G. P. (2000, noviembre). Neural networks for classification: A survey. *Trans. Sys. Man Cyber Part C*, *30*(4), 451–462. Descargado de `http://dx.doi.org/10.1109/5326.897072` doi: 10.1109/5326.897072

Zheng, R. T., Ngo, N. Q., Shum, P., Tjin, S. C., y Binh, L. N. (2005). A staged continuous tabu search algorithm for the global optimization and its applications to the design of fiber bragg gratings. *Computational Optimization and Applications*, *30*(3), 319–335. Descargado de `http://dx.doi.org/10.1007/s10589-005-4563-9` doi: 10.1007/s10589-005-4563-9