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), donde el proceso de muchas capas distribuidas en una arquitectura jerárquica 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). El DL 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). Se a mostrado haber producido bueno resultados para diversas tareas en 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).

En general, las técnicas del DL pueden clasificarse en modelos discriminativos profundos y modelos generativos (Deng y Yu, 2014). Ejemplos de modelos discriminativos son las redes neurales profundas (*Deep neural networks*, DNN), redes neuronales recurrentes (*Recurrent neural networks*, RNN) y redes neuronales convolucionales (*Convolutional neural networks*, CNN). Por otro lado, los modelos generativos, por ejemplo, son máquinas de Boltzmann restringidas (*Restricted Boltzmann machine*, RBMs), redes de creencias profundas (*Deep belief networks*, DBN), autocodificadores regularizados y máquinas profundas de Boltzmann (DBMs).

Las redes neuronales artificiales (*Artificial Neural Networks*, NN) han sido protagonistas de su propio renacimiento en el campo del ML con el surgimiento del DL (Bengio et al., 2006; G. E. Hin-

ton et al., 2006; Le et al., 2012; Ranzato et al., 2007). Las principales ideas detrás del nuevo enfoque abarcan una gama de algoritmos (Bengio y LeCun, 2007; G. E. Hinton et al., 2006), pero un principio en común es que una NN con múltiples capas ocultas, que la convierten en profundo, puede codificar características cada vez 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*, SGD), o una de sus variantes, para actualizar los pesos de la NN y de esa manera reducir el error total. Los descubrimientos en los últimos años han demostrado que, con suficientes datos de entrenamiento y con suficiente poder de procesamiento, el método de retropropagación y SGD resultan ser eficaces en la optimización de una NN de mucha profundidad y altamente conectada (Cireşan et al., 2012; He et al., 2015; Le et al., 2012). Esta realización ha llevado a registros sustantivos que se rompen en muchas áreas de las ML a través de la aplicación de la retropropagación en el aprendizaje profundo (Cireşan et al., 2012; He et al., 2015; Le et al., 2012), incluyendo el aprendizaje no supervisado (Bengio, 2009).

Las NN han sido ampliamente estudiadas y utilizadas en muchas aplicaciones de la inteligencia artificial. 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. El enfoque clásico para el entrenamiento de las 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 la dirección de descenso más pronunciada. 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 son la adaptación de un

término de momento (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 sin el uso de alguna heurística o alguna función adicional 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 sin evaluar la matriz Hessiana; 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 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.

Aunque el DL tiene buena reputación para resolver la tarea de aprendizaje, no es fácil el proceso de entrenamiento (Lamos-Sweeney, 2012). Recientes propuestas de técnicas de optimización han utilizado capas de pre-entrenamiento (G. E. Hinton y Salakhutdinov, 2006). Algunos ejemplos de los métodos exitosos para el entrenamiento de esta técnica son el Descenso de Gradiente Estocástico, Gradiente de Conjugado, Optimización Hessian y el Descenso de Subespacio de Krylov. Existen propuesta de algoritmos metaheurísticos que se pueden clasificar en base a la trayectoria (una sola solución) y basado en la población (Lamos-Sweeney, 2012).

1.2. DESCRIPCIÓN DEL PROBLEMA

La retropropagación basa su funcionamiento en multiplicaciones sucesivas basadas en el error para poder calcular los gradientes, y a medida que el error se propaga hacia la capa de entrada de la red él gradiente comienza a disminuír su valor por cada capa que atraviesa. Esto significa que el gradiente disminuirá de manera exponencial, lo que representa un problema para redes profundas, ya que las capas mas cercanas a la capa de entrada necesitarán más tiempo para ser entrenadas.

El método de aprendizaje basado en simmulated annealing permite la actualización de los pesos de la red sin mermar la capacidad de adaptación de los pesos. El método supone una alternativa efectiva 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.

1.3. SOLUCIÓN PROPUESTA

1.3.1. Características de la solución

Mediante el uso de el algoritmo *simulated annealing* se busca analizar la eficiencia que la NN alcanza en una red neuronal profunda frente a otros métodos de aprendizaje tales como SGD y RMSPROP.

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 distintas redes.

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.

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. Se analizará la misma arquitectura con diferentes reglas 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

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 se ven beneficiadas por el uso del algoritmo *simulated annealing* en la convergencia.
- Marco teórico: Una revisión de la literatura donde se aborda el problema planteado, para situarse en el contexto actual de los problemas. Se describirán redes neuronales 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.
- Análisis y verificación de los resultados: Los resultados se analizarán considerando los valores de convergencia de los distintos métodos.
- Presentación de los resultados: Se presentarán tablas que describan 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

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

Tabla 1.1: Especificaciones del equipo

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

2.1. ASPECTOS TEÓRICOS

2.1.1. Redes neuronales

El elemento básico de las NN es el nodo, que recibe un vector de entrada para producir una salida como muestra en la figura 2.1. Cada entrada tiene asociado un vector de pesos w, que se va modificando durante el proceso de aprendizaje. Cada unidad aplica una función f sobre la suma de las entradas ponderada por el vector de pesos como en la ecuacion 2.1.

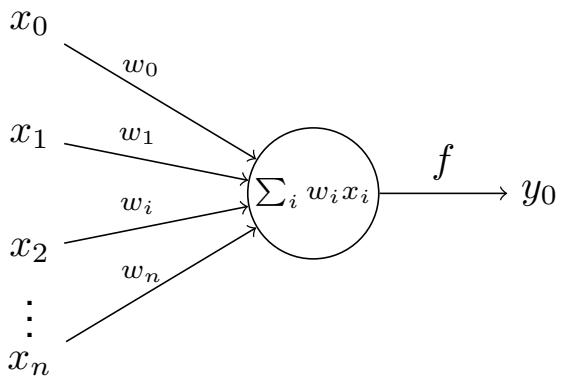

Figura 2.1: Perceptrón simple

$$y_i = \sum_j w_{ij} y_j \tag{2.1}$$

Donde el resultado puede servir como entrada de otras unidades.

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.

- 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.

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.

2.1.2. El perceptrón multicapa

El perceptrón multicapa es una generalización del perceptrón simple, y surgió como consecuencia de las limitaciones de dicha arquitectura para resolver problemas de clasificación no lineal. Marvin Minsky (1987) mostraron que el uso de varios perceptrones simples podía resultar una solución para resolver problemas no lineales. Sin embargo, dicha propuesta no presentaba una solución al problema de la actualización de los pesos de las capas, pues la regla de aprendizaje del perceptrón simple no era aplicable. Rummelhart, Hinton y Wilians presentaron la *regla delta generalizada*, una forma de propagar el error de la red desde la capa de salida hacia las capas anteriores.

Dentro de las redes neuronales, el perceptrón multicapa es una de las arquitecturas más usadas para resolver problemas. Esto es debido a que poseen la capacidad de ser un aproximador universal. Esto no implica que sea una de las redes más potentes o con mejores resultados, el perceptrón multicapa posee una serie de limitaciones, como el proceso de aprendizaje para problemas que dependan de un gran número de variables, la dificultad para realizar un análisis teórico de la red debido a la presencia de componentes no lineales y a la alta conectividad.

2.1.2.1. Arquitectura

El perceptron multicapa posee una estructura de capas compuestas por neuronas. Cada una de las capas está formada por un conjunto de neuronas y se distinguen tres tipos de capas: la capa de entrada, las capas ocultas y la capa de salida.

Las neurones de la capa de entrada se encargan de recibir los patrónes y propagar dichas señales a las neuronas de la capa siguiente. La última capa, la capa de salida, proporciona la respuesta de la red al patrón presentado. Las neuronas de las capas ocultas realizan el procesado de las

señales generadas por el patrón de entrada.

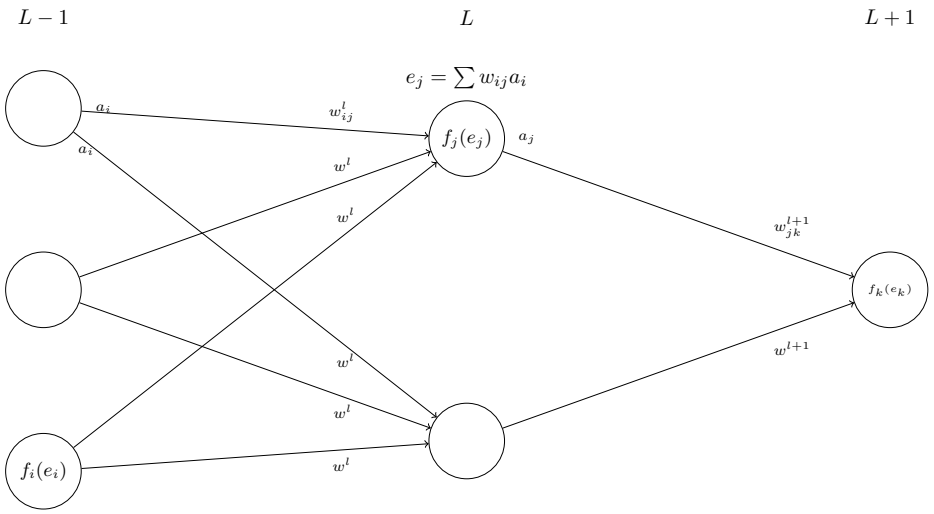

Figura 2.2: Neurona

En la figura 2.2 se observa que las conexiones van siempre hacia adelante, es decir, las neuronas de la capa l se conectan con las neuronas de la capa $l+1$. Las neuronas de una capa están conectadas a todas las neuronas de la capa siguiente.

2.1.2.2. Propagación de la entrada

El perceptrón multicapa define 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.

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, donde w_{ij}^c representa el peso de la conexion de la neurona i de la capa c. 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 de activación 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 de activación 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. Las funciones de activación mas utilizadas son la sigmoidal y la tangente hiperbólica, descritas en las escuaciones 2.2 y 2.3 respectivamente.

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

$$f_{tanh}(x) = \frac{1-\exp(-x)}{1+\exp(-x)} \tag{2.3}$$

Ambas funciones poseen como imagen intervalo de valores entre $[0, 1]$ y $[-1, 1]$ como se observa en la figura 2.3.

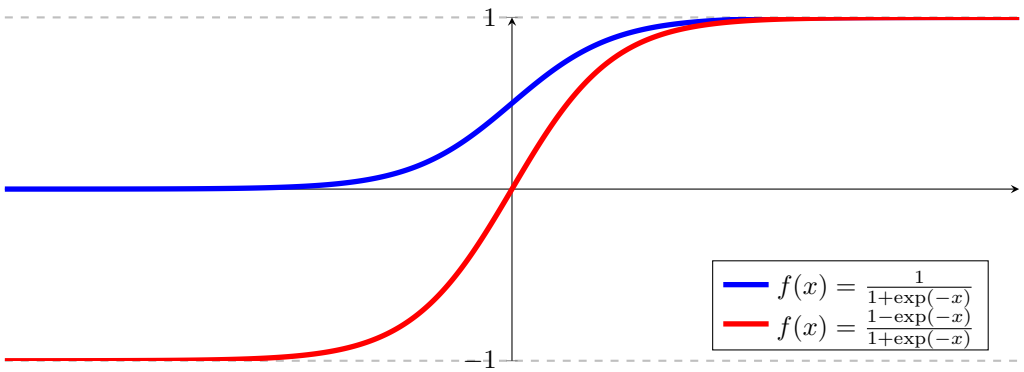

Figura 2.3: Funciones de activación mas utilizadas.

2.1.3. Algoritmo de retropropagación

Una regla de aprendizaje es el método que le permite adaptar los parámetros de la red. El perceptrón multicapa actualiza sus pesos en función de la salida obtenida de tal manera que los nuevos pesos permitan reducir el error de salida. Por tanto, para cada patrón de entrada a la red es necesario disponer de un patrón de salida deseada.

El objetivo es que la salida de la red sea lo más próxima posible a la salida deseada, debido a esto la es que el aprendizaje de la red se describe como un problema de minimización de la siguiente maner

$$\min_{W} E$$

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:

$$E = \frac{1}{N}\sum_{i=1}^{N} e(i) \tag{2.4}$$

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

$$e(i) = \frac{1}{n_C}\sum_{j=1}^{n_C}(s_j(i) - y^j(n))^2 \tag{2.5}$$

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.

Así es como el aprendizaje es equivalente a encontrar un mínimo de la función de error. 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 aminimización es un problema no lineal y se hace necesario el uso de técnicas de optización no lineales para su resolución.

Las técnicas 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 redes neuronales 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.

Si bien el aprendizaje de la red busca minimizar el error total de la red, el procedimiento está basado en métodos del agradiente estocástico, que son una sucesión de minimizaciones del error en función de cada patrón $e(i)$, en lugar de minimizar el error total E de la red. Aplicando el método del gradiente estocástico, cada parámetro w se modifica para cada patrón de entrada n según la siguiente regla de aprendizaje

$$w(i) = w(n-1) - \alpha \frac{\partial e(i)}{\partial w} \tag{2.6}$$

donde $e(i)$ es el error para el patrón de entrada i dado por la ecuación 2.5, y α es la tasa de aprendizaje, éste último determina el desplazamiento en la superficie del error.

Como las neuronas están ordenadas por capas y en distintos niveles, es posible aplicar el método del gradiente de forma eficiente, resultando en el *algoritmo de retropropagación* (Rumelhart et al., 1986a) o *regla delta generalizada*. 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 algoritmo de retropropagación es el método de entrenamiento más utilizado en redes con conexión hacia adelante. Es un método de aprendizaje supervisado de gradiente descendente, en el que se distinguen claramente dos fases:

1. Se aplica un patrón de entrada, el cual se propaga por las distintas capas que componen la red hasta producir la salida de la misma. Esta salida se compara con la salida deseada y se calcula el error cometido por cada neurona de salida.

2. Estos errores se transmiten desde la capa de salida, hacia todas neuronas de las capas anteriores (Fritsch, 1996). Cada neurona recibe un error que es proporcional a su contribución sobre el error total de la red. Basándose en el error recibido, se ajustan los errores de los pesos sinápticos de cada neurona.

2.1.4. El gradiente descendente

2.1.4.1. El desvanecimiento del gradiente

El problema del gradiente desvaneciente nace en las NN profundas, éstas utilizan funciones cuyo gradiente tienden a estar entre 0 y 1. Debido a que estos gradientes pequeños se multiplican durante la retropropagación, tienden a *desvanecerse* a través de las capas, evitando que la red aprenda en redes muy profundas.

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

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

es $y^i(t) = f_i(net_i(t))$. Además w_{ij} es el peso de la conexión desde la unidad j hasta la unidad i, $d_k(t)$ 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.

En un tiempo arbitrario $\tau \leq t$, el error de la señal en una unidad j, que no sea de entrada, será la suma de los errores externos y la señal propagada anterioremente como muestra la ecuación 2.7.

$$\vartheta(\tau) \quad = \quad f_j'(net_j(\tau)) \left(E_j(\tau) + \sum_i w_{ij} \vartheta_i(\tau+1) \right) \tag{2.7}$$

Entonces, los pesos actualizados serán

$$w_{jl}^{new} = w_{jl}^{old} + \alpha\vartheta_j(\tau)y^l(\tau - 1)$$

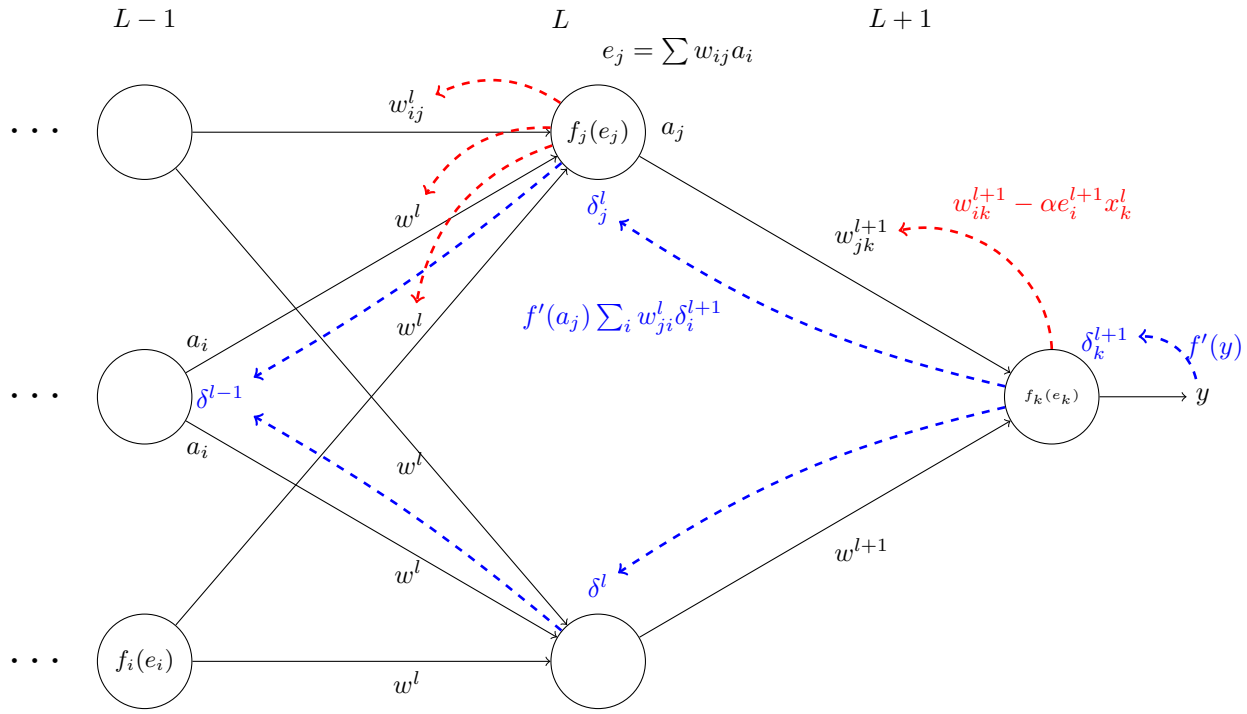

Figura 2.4: Gradiente descendente

2.1.5. Simulated Annealing

Harland1988

2.2. REVISIÓN DE LA LITERATURA

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

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

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

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.

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

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

Cireşan, D., Meier, U., Masci, J., y Schmidhuber, J. (2012). Multi-column deep neural network for traffic sign classification. *Neural Networks*, *32*, 333 - 338. Descargado de http://www.sciencedirect.com/science/article/pii/S0893608012000524 (Selected Papers from {IJCNN} 2011) doi: https://doi.org/10.1016/j.neunet.2012.02.023

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

Deng, L., y Yu, D. (2014). Deep learning: Methods and applications. *Foundations and Trends® in Signal Processing*, *7*(3–4), 197-387. Descargado de http://dx.doi.org/10.1561/2000000039 doi: 10.1561/2000000039

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

Fritsch, J. (1996). *Modular neural networks for speech recognition* (Masters Thesis). KIT.

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

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

He, K., Zhang, X., Ren, S., y Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, *abs/1512.03385*. Descargado de http://arxiv.org/abs/1512.03385

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

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

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

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

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

Marvin Minsky, S. A. P. (1987). *Perceptrons: An introduction to computational geometry* (Expanded ed.). The MIT Press.

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/

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`

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.

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

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

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

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