

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

Propuesta de Tesis

Informe

Nombre: Felipe Alberto Reyes González
Programa: Magíster en Ingeniería Informática
Profesor patrocinante: Victor Parada
Cel.: 890 26 317
email: felipe.reyesg@usach.cl

23 de abril de 2017

Tabla de contenido

1. Introducción	3
1.1. El perceptrón simple	3
1.2. Las redes neuronales	4
1.3. El perceptrón multicapa	4
1.3.1. Arquitectura	5
1.3.2. Propagación de la entrada	6
2. Algoritmo de retropropagación	9
3. El gradiente descendente	9
3.1. El desvanecimiento del gradiente	11
Bibliografía	12

Índice de tablas

Índice de figuras

1. Perceptrón simple	3
2. Neurona	6
3. Funciones de activación mas utilizadas.	8
4. W_{ij}^l es el peso de la n -ésima neurona en la capa $l - 1$ a la j -ésima neurona de la capa l de la red.	10

1 Introducción

El primer modelo de red neuronal artificiales (NN) fue propuesto en 1943 por McCulloch y Pitts en términos de un modelo computacional de actividad nerviosa. Las NN se han inspirado en las redes neuronales biológicas y las conexiones que construyen. Este modelo era un modelo binario, donde cada neurona posee un umbral, y sirvió de base para los modelos posteriores.

Las características principales de las NN son las siguientes:

1. Auto-organización y adaptabilidad: utilizan algoritmos de aprendizaje adaptativo y auto-organizativo, por lo que ofrecen mejores posibilidades de procesamiento robusto y adaptativo.
2. Procesado no lineal: aumenta la capacidad de la red para aproximar funciones, clasificar patrones y aumenta su inmunidad frente al ruido.
3. Procesado paralelo: normalmente se usa un gran número de nodos de procesamiento, con alto nivel de interconectividad.

1.1 El perceptrón simple

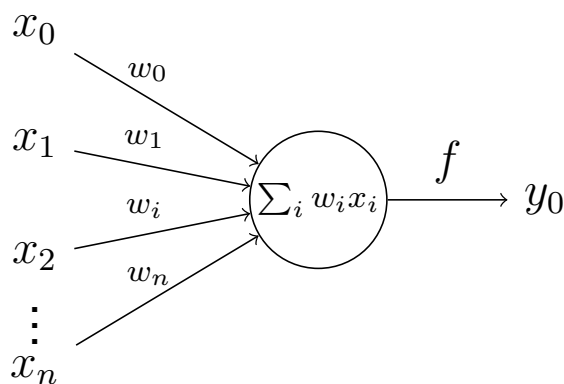


Figura 1: Perceptrón simple

1.2 Las 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 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

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

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.

1.3 El perceptrón multicapa

El perceptrón multicapa es una generalización del perceptrón simple, y surgió como consecuencia de las limitaciones de dicha arquitectura para resol-

ver problemas de clasificación no lineal. Minsky y Papert (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.

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.

1.3.1 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 neuronas de la capa de entrada se encargan de recibir los patrones 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 procesamiento de las señales generadas por el patrón de entrada.

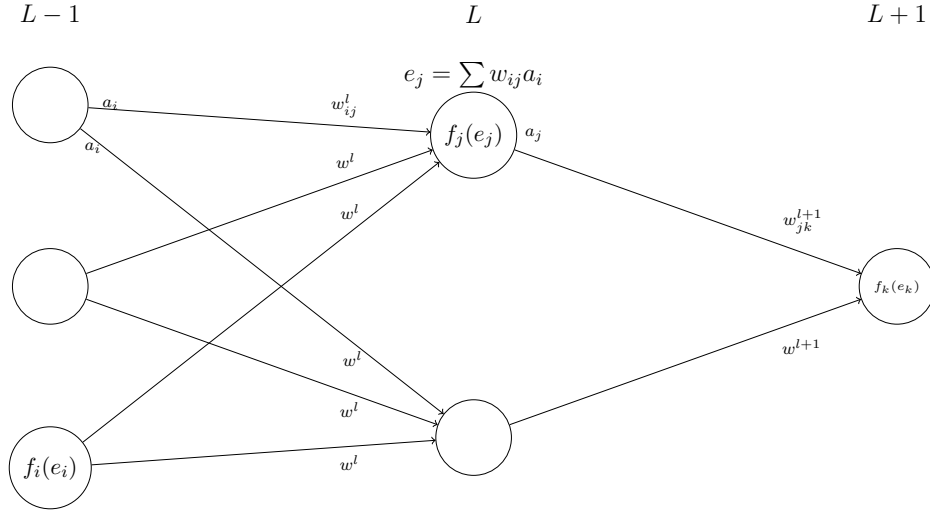


Figura 2: Neurona

En la figura 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.

1.3.2 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 conexión 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 encar-

gan de transmitir la entrada recibida, por lo tanto

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

donde $X = (x_1, x_2, \dots, x_n)$ 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

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

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

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

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

La función f es la función de activación de la neurona. Las funciones de activación mas utilizadas son la sigmoideal y la tangente hiperbólica (ver ecuación 1 y 2 respectivamente).

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

$$f_{tanh}(x) = \frac{1 - \exp(-x)}{1 + \exp(-x)} \quad (2)$$

Ambas funciones poseen como imagen intervalo de valores entre $[0, 1]$

y $[-1, 1]$ como se observa en la figura 3.

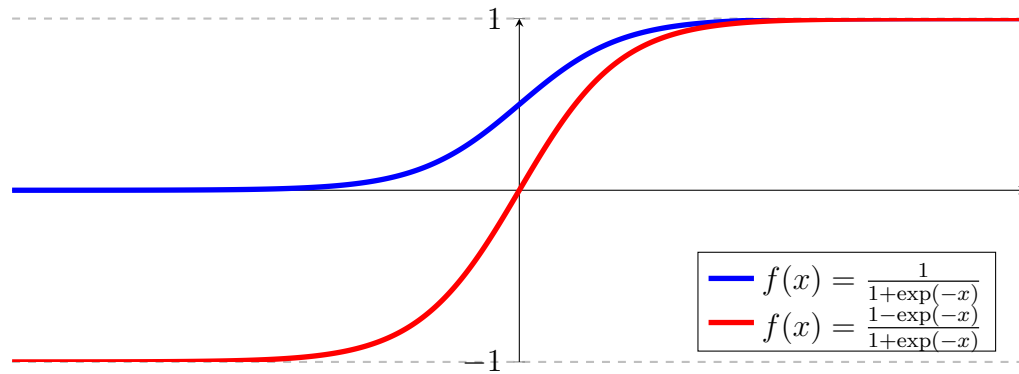


Figura 3: Funciones de activación mas utilizadas.

2 Algoritmo de retropropagación

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.

3 El gradiente descendente

El gradiente descendente busca los punto $p \in \Omega$ donde funciones del tipo $f : \Omega \subseteq \mathbb{R}^m \rightarrow \mathbb{R}$ alcanzan su mínimo. La idea de este método se basa en que si f es una función diferenciable en todo su dominio Ω , entonces la derivada de f es un punto $p \in \Omega$ en dirección de un vector unitario $v \in \mathbb{R}^m$ se define como

$$df_p(v) = \nabla f(p)v$$

Observe que la magnitud de la ecuación es

$$|df_p(v)| = ||\nabla f(p)|| ||v|| \cos \theta = ||\nabla f(p)|| \cos \theta$$

Dicha magnitud es máxima cuando $\theta = 2n\pi, n \in \mathbb{Z}$. Es decir, para que

$|df_d(v)|$ sea máxima, los vectores $\nabla f(p)$ y v debe ser paralelo. De esta manera, la función f crece más rápidamente en la dirección del vector $\nabla f(p)$ y decrece más rápidamente en la dirección del vector $-\nabla f(p)$. Dicha situación sugiere que la dirección negativa del gradiente $-\nabla f(p)$ es una buena dirección de búsqueda para encontrar el minimizador de la función f .

Sea $f : \Omega \subseteq \mathbb{R} \rightarrow \mathbb{R}$, si f tiene un mínimo en p , para encontrar a p se construye una sucesión de punto $\{p_t\}$ tal que p_t converge a p . Para resolver esto, comenzamos en p_t y nos desplazamos una cantidad $-\lambda_t \nabla f(p_t)$ para encontrar el punto p_{t+1} más cercano a p , es decir:

$$p_{t+1} = p_t - \lambda_t \nabla f(p_t)$$

donde λ_t se selecciona de tal manera que $p_{t+1} \in \Omega$ y $f(p_t) \geq f(p_{t+1})$

El parámetro λ_t se seleccionara para maximizar la cantidad a la que decrece la función f en cada paso.

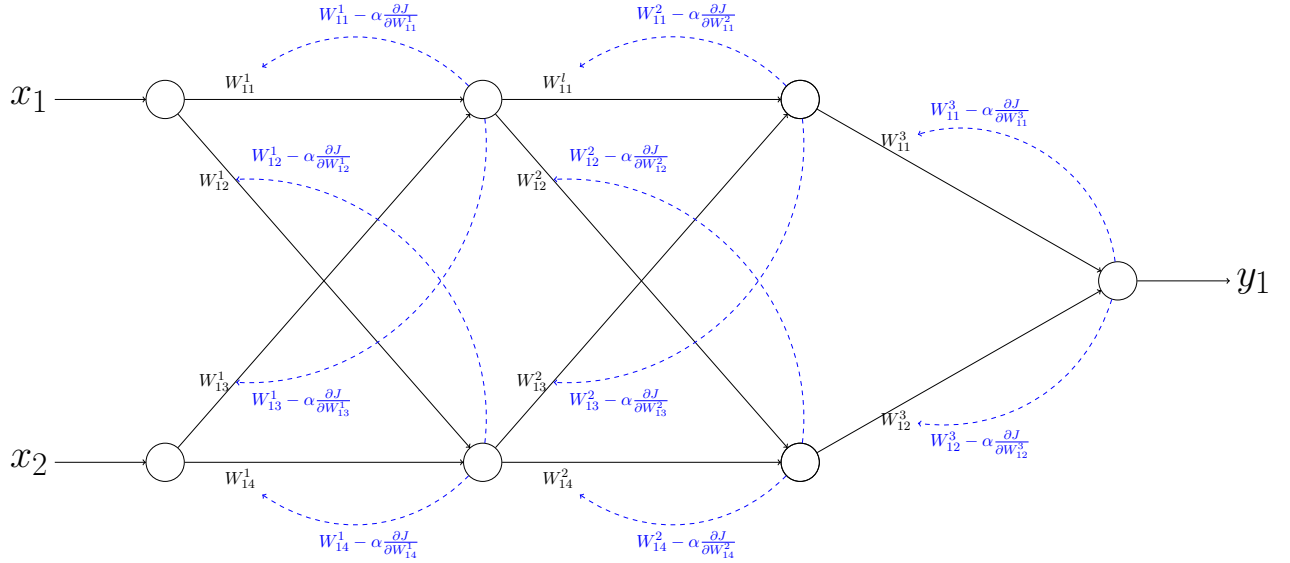


Figura 4: W_{ij}^l es el peso de la n -ésima neurona en la capa $l - 1$ a la j -ésima neurona de la capa l de la red.

3.1 El desvanecimiento del gradiente

Bibliografía

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