Redes Neuronales Multicapa I

Modelos de la computación (Aprendizaje supervisado)

Francisco Fernández Navarro

Departamento de Lenguajes y Ciencias de la Computación
Área: Ciencias de la Computación e Inteligencia Artificial

Índice de contenidos

Introducción

- Interés en investigación en redes multicapa desde Rosenblatt (1962) y Madalines de Widrow (1962).
- Perceptrón simple resuelve problemas de clasificación y funciones lógicas, pero no XOR (ni problemas linealmente no separables).
- Un Perceptrón de dos capas puede implementar XOR.

Ejemplo red multicapa problema XOR

Si deseamos implementar la función XOR basta con utilizar dos unidades de proceso en la capa oculta.

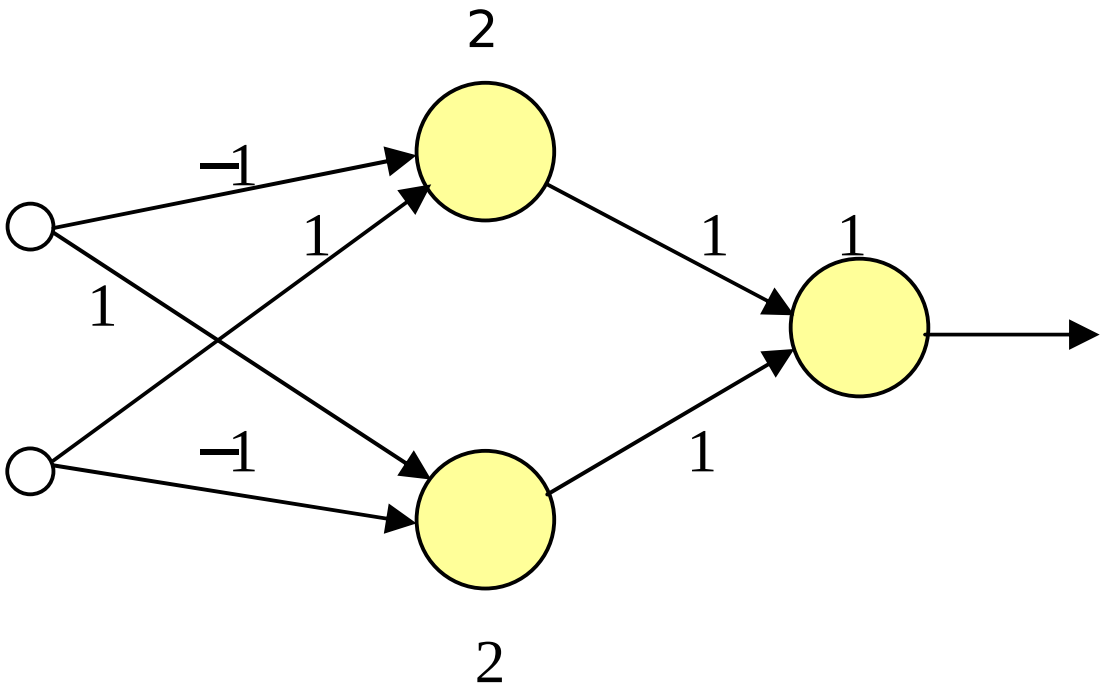
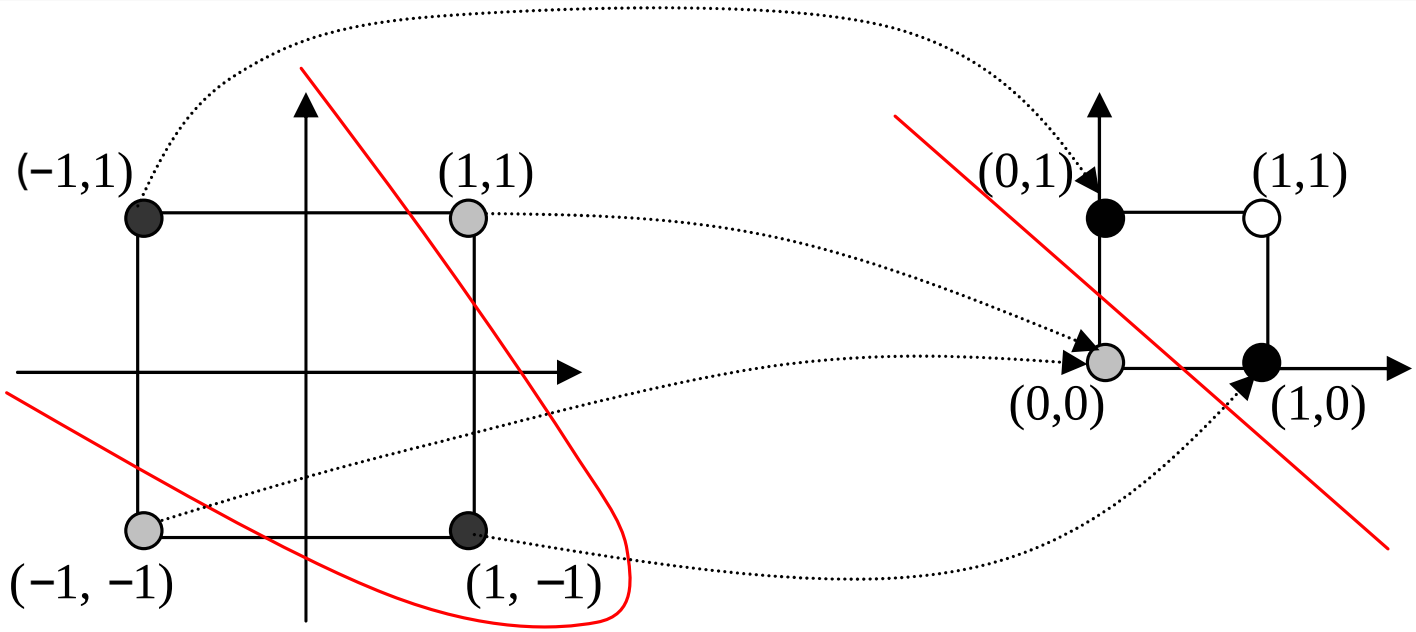


Figure: Implementación del problema XOR con una red multicapa con dos neuronas en capa oculta

Introducción

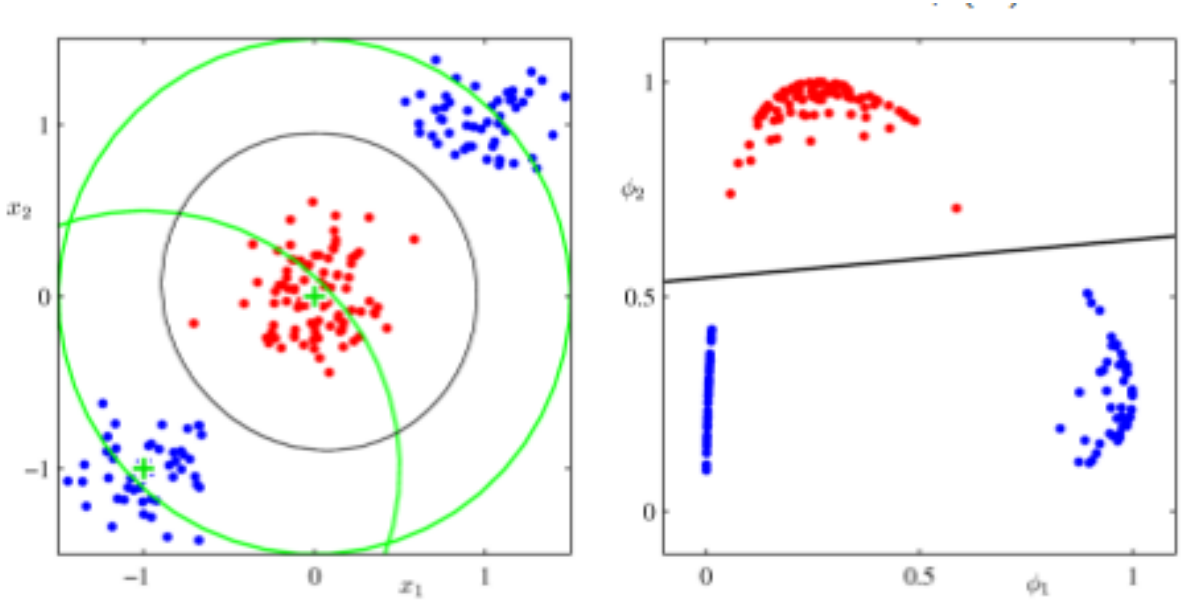
¿Qué papel desempeña la capa intermedia?

La capa intermedia realiza una proyección de los patrones de entrada en un cubo cuya dimensión viene dada por el número de unidades de la capa oculta. Se trata de realizar una proyección en la que resulten separables linealmente los patrones de entrada de manera que la unidad de salida se pueda realizar una clasificación correcta.



Introducción

Problema de clasificación binario con dos atributos de entrada (x_1 y x_2). Un ejemplo de problema no separable linealmente en el espacio de atributos originales. Se utilizan dos neuronas en capa oculta (cuya salida en el problema está denotada como ϕ_1 y ϕ_2).



Índice de contenidos

Modelo funcional

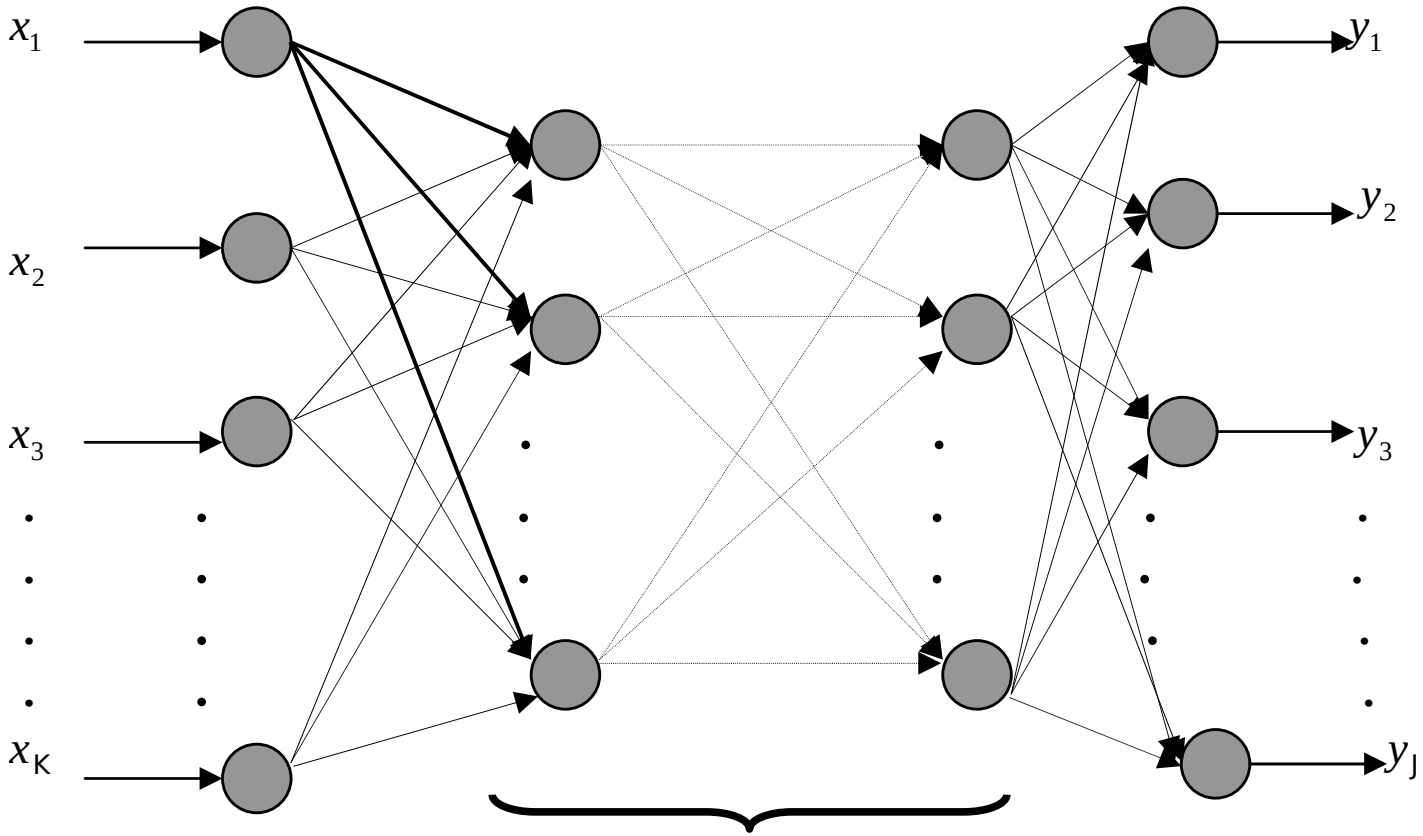
Componentes de una red neuronal multicapa

- Una capa de entrada (definida por el número de atributos del problema, K).
- Una o varias capas con neuronas en capa oculta.
- Una capa de salida con tantas neuronas como clases o clases -1 tenga asociado el problema (denotadas como J)

¿Porqué podemos no estimar una de las clases

En un contexto probabilístico, el valor de la última clase puede estimarse como 1 menos la suma del resto de probabilidades.

Ejemplo de perceptrón multicapa



Nomenclatura

Clasificación multi-clase

Los parámetros de los modelos de aprendizaje se estiman a partir de un conjunto de entrenamiento $\mathcal{D} = (\mathbf{X}, \mathbf{Y}) = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$, donde $\mathbf{x}_n = (x_{n1}, x_{n2}, \ldots, x_{nK}) \in \mathbb{R}^K$ es el vector de atributos del n-ésimo patrón, K es la dimensión del espacio de entrada (número de atributos en el problema), $\mathbf{y}_n \in \{0,1\}^J$ es la etiqueta de clase asumiendo la codificación "1-de-J" ($y_{nj} = 1$ si $\boldsymbol{x}_n$ es un patrón de la j-ésima clase, $y_{nj} = 0$ en caso contrario), y J es el número de clases. Denotemos $\mathbf{Y}$ como $\mathbf{Y} = (\mathbf{Y}_1 \ldots \mathbf{Y}_J) = \begin{pmatrix} \mathbf{y}_1' \\ \vdots \\ \mathbf{y}_N' \end{pmatrix} \in \{0,1\}^{N \times J}$, donde $\mathbf{Y}_j$ es la j-ésima columna de la matriz $\mathbf{Y}$. La función $f : \mathbb{R}^K \to \{0,1\}^J$.

Nomenclatura

Modelo funcional

La salida del modelo para un patrón n-ésimo, $\mathbf{x}_n \in \mathbb{R}^K$ es:

$$\hat{y}_{nj} = \mathrm{g}_1\left(\sum_{l=1}^{L} w_{lj} s_{ln}\right) = \mathrm{g}_1\left(\sum_{l=1}^{L} w_{lj}\left(\mathrm{g}_2\left(\sum_{k=1}^{K+1} t_{lk} x_{nk}\right)\right)\right). \quad (1)$$

Definición de variables

- $\hat{y}_{nj}$: Estimación de salida de la clase j para el patrón n.
- s_{ln}: Salida de la neurona en capa oculta j para el patrón n.
- L: Número de neuronas en la capa oculta.
- $\mathrm{g}_1(.)$ y $\mathrm{g}_2(.)$: Funciones de transferencia de capa de salida y oculta.
- w_{lj}: Peso sináptico neurona salida j y neurona oculta l.
- t_{lk}: Peso sináptico neurona oculta l y entrada k.

Nomenclatura

Importante

El modelo de base que estudiaremos en el tema no consideraremos, por simplicidad, el sesgo en la capa de salida. En el caso de que se incluyera la salida del modelo para un patrón n-ésimo, $\mathbf{x}_n \in \mathbb{R}^K$ sería:

$$\hat{y}_{nj} = \mathrm{g}_1\left(\sum_{l=1}^{L} w_{lj} s_{ln} - \theta_j\right) = \mathrm{g}_1\left(\sum_{l=1}^{L} w_{lj}\left(\mathrm{g}_2\left(\sum_{k=1}^{K+1} t_{lk} x_{nk}\right)\right) - \theta_j\right), \quad (2)$$

donde θ_j es el umbral del nodo de salida j-ésimo.

Sesgo en la capa de salida

¿Que función tiene el sesgo en la capa de salida?

Applied Soft Computing 133 (2023) 109914

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

A multi-class classification model with parametrized target outputs for randomized-based feedforward neural networks

Antonio Manuel Durán-Rosal [a], Aggeo Durán-Fernández [b], Francisco Fernández-Navarro [c,*], Mariano Carbonero-Ruz [a]

[a] Department of Quantitative Methods, Universidad Loyola Andalucía, Spain
[b] U-tad Centro Universitario de Tecnología y Arte Digital, Madrid, Spain
[c] Department of Computer Languages and Computer Science, University of Málaga, Spain

Índice de contenidos

Problema de optimización

Problema de optimización

Los parámetros del modelo se estiman a través del siguiente problema de optimización:

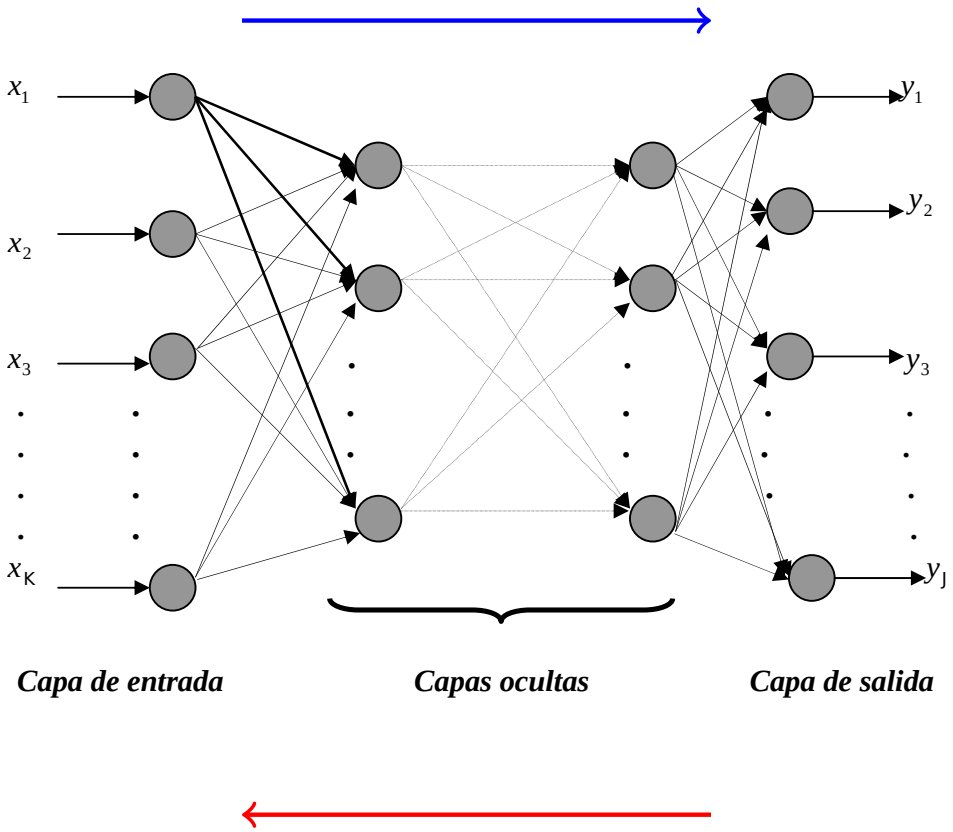
$$\min_{\mathbf{w}\in\mathbb{R}^{L\times J},\mathbf{t}\in\mathbb{R}^{L\times(K+1)}} \quad E(\mathbf{w},\mathbf{t}) = \frac{1}{2}\sum_{j=1}^{J}\sum_{n=1}^{N}(y_{nj}-\hat{y}_{nj})^2,$$

donde $\mathbf{w} \in \mathbb{R}^{L\times J}$ es la matriz de pesos de capa oculta y $\mathbf{t} \in \mathbb{R}^{L\times(K+1)}$ la matriz de pesos de capa de entrada.

Índice de contenidos

Estimación de parámetros

Cálculo de la salida: Forward



Retropropagación del error: Backward

Nomenclatura

El procedimiento será iterativo ($i = 1, \ldots, I$) y las modificaciones se realizarán de patrón a patrón ($1 \leq n \leq N$), por lo que será necesario definir:

- $\hat{y}_{nj}(i)$: Estimación de salida de la clase j para el patrón n, en la iteración i.
- $s_{ln}(i)$: Salida de la neurona en capa oculta l para el patrón n, en la iteración i.
- $w_{lj}(i, n)$: Peso sináptico neurona salida j y neurona oculta l, en la iteración i (patrón n).
- $t_{lk}(i, n)$: Peso sináptico neurona oculta l y entrada k (iteración i, patrón n).

Además por facilitar la comprensión definiremos:

$$h_{jn}(i) = \sum_{l=1}^{L} w_{lj} s_{ln}(i), u_{ln}(i) = \sum_{k=1}^{K} t_{lk} x_{nk} \tag{3}$$

por lo que $\hat{y}_{nj}(i) = \mathrm{g}_1\left(h_{jn}(i)\right)$

Pesos de capa oculta a capa de salida

Estimación $w_{lj}, j = \{1, \ldots, J\}, l = \{1, \ldots, L\}$

La regla de modificación de los pesos sinápticos de la capa de salida será:

$$w_{lj}(i, n+1) = w_{lj}(i, n) + \Delta w_{lj}(i, n), \tag{4}$$

donde

$$\Delta w_{lj}(i, n) = -\eta \frac{\partial E(\mathbf{w}, \mathbf{t})}{\partial w_{lj}(i, n)} = \eta (y_{nj} - \hat{y}_{nj}(i)) g_1'(h_{jn}(i)) s_{ln}(i). \tag{5}$$

Llamamos $\delta_j^2(i, n) = (y_{nj} - \hat{y}_{nj}(i)) g_1'(h_{jn}(i))$, por lo que

$$\Delta w_{lj}(i, n) = \eta \delta_j^2(i, n) s_{ln}(i). \tag{6}$$

Pesos de capa oculta a capa de entrada

Estimación $t_{lk}, l = \{1, \ldots, L\}, k = \{1, \ldots, K\}$,

La regla de modificación de los pesos sinápticos de la capa oculta será:

$$t_{lk}(i, n+1) = t_{lk}(i, n) + \Delta t_{lk}(i, n),$$

donde

$$\begin{aligned}
\Delta t_{lk}(i,n) &= -\eta \frac{\partial E(\mathbf{w}, \mathbf{t})}{\partial s_{ln}(i)} \frac{\partial s_{ln}(i)}{\partial t_{lk}(i)} \\
&= \eta \sum_{j=1}^{J} (y_{nj} - \hat{y}_{nj}(i)) g_1'(h_{jn}(i)) w_{lj}(i,n) g_2'(u_{ln}(i)) x_{nk} \\
&= \eta \sum_{j=1}^{J} \delta_j^2(i,n) w_{lj}(i,n) g_2'(u_{ln}(i)) x_{nk}
\end{aligned}$$

Pesos de capa oculta a capa de entrada

Estimación $t_{lk}, l = \{1, \ldots, L\}, k = \{1, \ldots, K\},$

Llamamos $\delta_l^1(i,n) = g_2'(u_{ln}(i)) \sum_{j=1}^{J} \delta_j^2(i,n) w_{lj}(i,n)$, por lo que

$$\Delta t_{lk}(i,n) = \eta \delta_l^1(i,n) x_{nk}. \tag{7}$$

Derivadas de las funciones de transferencias

Tipos de función de transferencia

- La función logística: $g(x) = \frac{1}{1+\exp(-2\beta x)}$. Derivada muy simple: $g'(x) = 2\beta g(x)(1 - g(x))$.
- La función tangente hiperbólica: $g(x) = \tanh(\beta x) = \frac{e^{\beta x} - e^{-\beta x}}{e^{\beta x} + e^{-\beta x}}$. Derivada muy simple: $g'(x) = \beta(1 - g(x)^2)$

Índice de contenidos

Entrenamiento por lotes

- Se introducen los N patrones simultáneamente y se evalúan las salidas de la red comparándolas con las salidas deseadas.
- En este caso, los pesos sinápticos no dependen de n pues se cambian solo después de evaluar todos los patrones.
- En este caso tomamos también la función de error total, pero la dividiremos por N para interpretarla como error cuadrático medio por patrón.

Problema de optimización

Problema de optimización

Los parámetros del modelo se estiman a través del siguiente problema de optimización:

$$\min_{\mathbf{w}\in\mathbb{R}^{L\times J},\mathbf{t}\in\mathbb{R}^{L\times(K+1)}} \quad E(\mathbf{w},\mathbf{t}) = \frac{1}{2N}\sum_{j=1}^{J}\sum_{n=1}^{N}(y_{nj}-\hat{y}_{nj})^2,$$

donde $\mathbf{w} \in \mathbb{R}^{L\times J}$ es la matriz de pesos de capa oculta y $\mathbf{t} \in \mathbb{R}^{L\times(K+1)}$ la matriz de pesos de capa de entrada.

Problema de optimización

Problema de optimización

La variación de cada parámetro vendrá definida como:

$$\Delta w_{lj}(i) = \frac{\eta}{N}\sum_{n=1}^{N}(y_{nj} - \hat{y}_{nj}(i))g_1'(h_{jn}(i))s_{ln}(i)$$

$$\Delta t_{lk}(i) = \frac{\eta}{N}\sum_{n=1}^{N}\sum_{j=1}^{J}(y_{nj} - \hat{y}_{nj}(i))g_1'(h_{jn}(i))w_{lj}(i)g_2'(u_{ln}(i))x_{nk}$$

Índice de contenidos

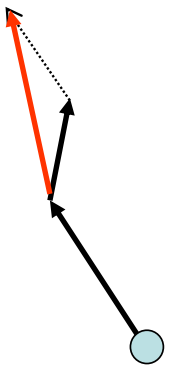
Regla de aprendizaje con Momentos

Idea: Hinton y Williams (1986) propusieron que se tuviera en cuenta en la variación de un parámetro también el gradiente de la iteración anterior y realizar un promedio con el gradiente de la iteración actual. De esta forma, los pesos de capa de salida son modificados en función de la siguiente ecuación:

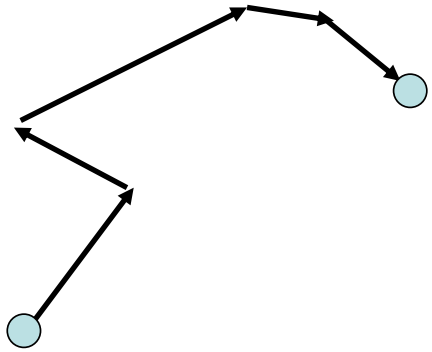
$$\Delta w_{lj}(i, n+1) = \alpha \Delta w_{lj}(i, n) + \eta \delta_j^2(i, n) s_{ln}(i),$$

con $0 \leq \alpha < 1$.
Dicho promedio produce un efecto de reducción drástica de las fluctuaciones del gradiente en iteraciones consecutivas, puesto que con frecuencia se produce cierto zigzagueo en el descenso por el gradiente que hace que el algoritmo sea lento (poco eficiente).

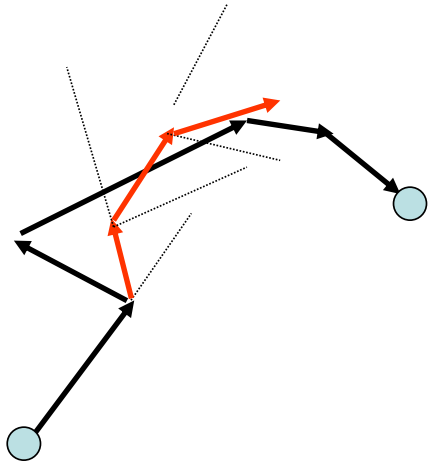
Regla de aprendizaje con Momentos



Aprendizaje con momentos
->Efecto acumulador



Aprendizaje normal
(sin momentos)



Aprendizaje con momentos
Caso de oscilaciones
Efecto estabilizador

Índice de contenidos

Retropropagación estándar (Backpropagation):

- Descenso de gradiente: Utiliza el descenso de gradiente para ajustar los pesos de la red. Calcula el gradiente de la función de costo con respecto a los pesos y luego actualiza los pesos en la dirección opuesta al gradiente.
- Limitaciones: Puede ser sensible a problemas de convergencia lenta o estancamiento en óptimos locales. También puede sufrir de problemas de desvanecimiento o explosión de gradientes en redes profundas.
- El estudiado en la asignatura.

Quickprop:

- Adaptación de tasa de aprendizaje: Quickprop es un algoritmo que ajusta dinámicamente la tasa de aprendizaje durante el entrenamiento. Cuando el error disminuye, Quickprop acelera la tasa de aprendizaje, lo que puede ayudar a converger más rápido.
- Limitaciones: Aunque puede acelerar el entrenamiento en algunos casos, puede ser más sensible a la elección de hiperparámetros y menos estable en comparación con otros algoritmos más modernos.

¡Gracias por vuestra atención!