Sistemas de Inteligencia artificial
Perceptron multicapa

Rodrigo Ramele, Juliana Gambini,
Alan Pierri, Juan Santos,
Paula Oseroff, Eugenia Piñeiro,
Pedro Momesso

2021

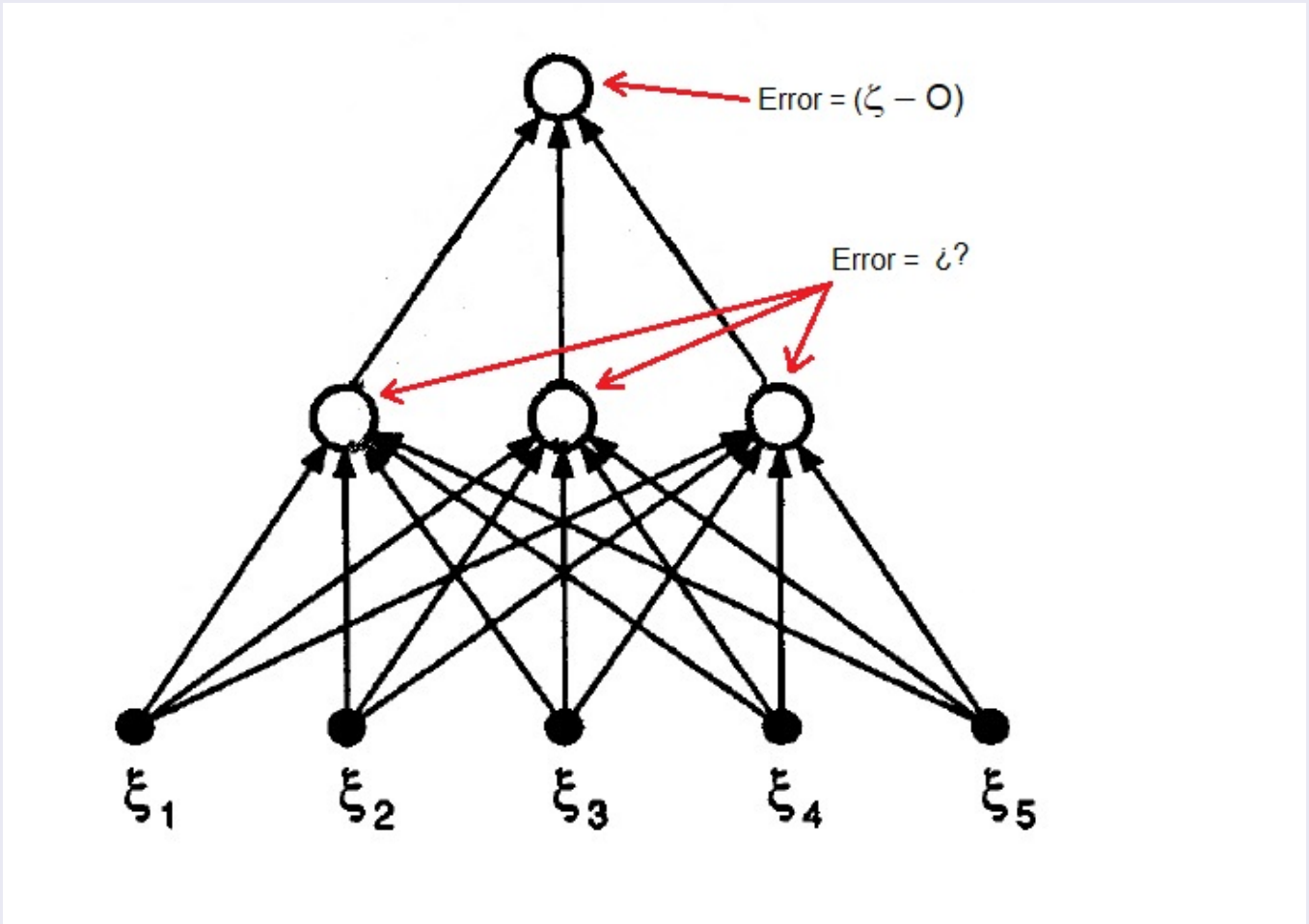
Redes neuronales artificiales

Perceptron multicapa

- Aprender en un perceptron simple es responder a la pregunta ¿cómo actualizo los pesos cuando la salida es distinta a la salida esperada?
- Se toma alguna proporción del error y con eso se corrigen los pesos de la conexiones que unen directamente la entrada con las salidas.
- Pero ¿cómo hacerlo en una arquitectura como la que sigue?

Redes neuronales artificiales

Perceptron multicapa



Redes neuronales artificiales

Perceptron multicapa

- En base al error en la unidad de salida podría corregir los pesos que unen la capa del medio con la unidad de salida, pero ahora ¿cúal es el error en las unidades de la capa del medio?
- Esto es, qué estado de activación deberían tener (deseado) las unidades de la capa del medio para poder establecer un error y de acuerdo a eso corregir los pesos de las conexiones que unen las entradas con las unidades de las capas del medio?

Redes neuronales artificiales

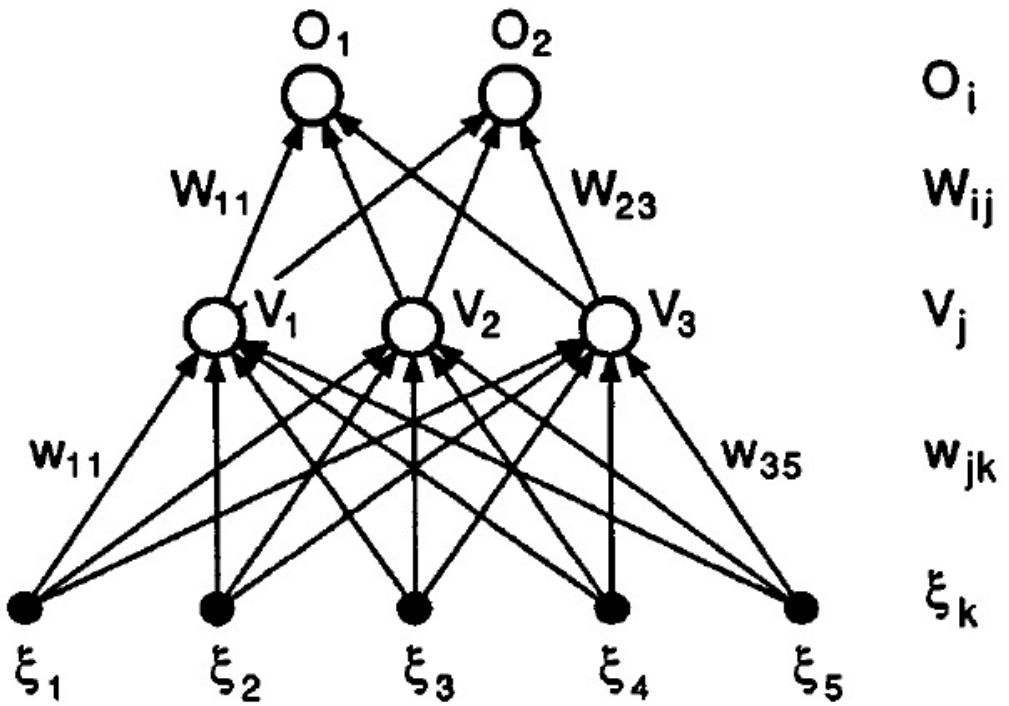
Perceptron multicapa

- En 1986, Rumelhart, Hinton, y Williams con su trabajo de PDP (Procesamiento paralelo distributivo) hicieron un importante aporte en este sentido.
- Hubo varios investigadores que trabajaron en la misma dirección tales como Le Cun (1985), Parker (1985), Werbos (1974), entre otros.

Redes neuronales artificiales

Perceptron multicapa

Consideremos una red multicapa con 2 capas.



Redes neuronales artificiales

Perceptron multicapa

- Llamaremos a la capa del medio como **capa oculta**.
- Las conexiones entre los nodos de entrada y la capa oculta los vamos a notar con w minúscula, mientras que los pesos de las conexiones entre la capa oculta y la de salida con W mayúscula.
- El estado de activación de las unidades en la capa oculta lo notaremos con V mayúscula.

Redes neuronales artificiales

Perceptron multicapa

- El índice i refiere a las unidades de salida, j a las unidades ocultas y k a las unidades de entrada.
- El supraíndice μ indica el número de ejemplo (o patrón) y varía entre 1 y p, donde p es la cantidad de ejemplos en el conjunto de entrenamiento.
- El subíndice k indica el elemento del ejemplo y varía entre 1 y N, donde N es la dimensión de cada ejemplo.

Redes neuronales artificiales

Perceptron multicapa

- Presentado un ejemplo μ a la red, cada unidad en la capa oculta alcanza un estado de exitación h_j^μ dado por:

$$h_j^\mu = \sum_k w_{jk} \xi_k^\mu$$

lo cual produce un estado de activación de las unidades de la capa oculta V_j^μ:

$$V_j^\mu = g(h_j^\mu) = g(\sum_k w_{jk} \xi_k^\mu)$$

Redes neuronales artificiales

Perceptron multicapa

- Cada unidad de salida alcanza un estado de exitación h_i^μ dado por:

$$h_i^\mu = \sum_j W_{ij} V_j^\mu = \\ = \sum_j W_{ij} g(\sum_k w_{jk} \xi_k^\mu)$$

lo cual produce un estado de activación de las unidades de salida O_i^μ:

$$O_i^\mu = g(h_i^\mu) = g(\sum_j W_{ij} V_j^\mu) = \\ = g(\sum_j W_{ij} g(\sum_k w_{jk} \xi_k^\mu))$$

Perceptron multicapa

- Si consideramos la función de costo que ya habíamos visto para el perceptron simple

$$E(w) = \frac{1}{2}\sum_{\mu,i}(\zeta_i^\mu - O_i^\mu)^2,$$

podemos escribirla ahora como

$$E(w) = \frac{1}{2}\sum_{\mu,i}(\zeta_i^\mu - g(\sum_j W_{ij} g(\sum_k w_{jk}\xi_k^\mu)))^2.$$

Perceptron multicapa

- Como se puede observar, $E()$ depende de W_{ij} y de w_{jk} y además es derivable.
- Ahora sí podemos obtener una forma de actualizar los pesos de las conexiones w_{jk} entre las unidades de entrada y las unidades de la capa oculta.
- **No hace falta conocer explícitamente** un estado de activación deseado para cada una de las unidades en la capa oculta.

Redes neuronales artificiales

Perceptron multicapa

- La táctica que implica este enfoque es 'retropropagar' el error $(\zeta_i^\mu - O_i^\mu)$ de cada unidad de la capa de salida hacia las capas inferiores, en este caso, la capa oculta.
- A partir de esto, se podrán actualizar los pesos de las conexiones w_{jk}.

Redes neuronales artificiales

Perceptron multicapa

- Recordemos la expresión del costo:

$$E(w) = \frac{1}{2}\sum_{\mu}\sum_{i}(\zeta_i^{\mu} - O_i^{\mu})^2 = \frac{1}{2}\sum_{\mu}\sum_{i}(\zeta_i^{\mu} - g(\sum_{j} W_{ij} V_j^{\mu}))^2.$$

- Nuevamente, usando la técnica del gradiente descendente, podemos primero calcular la actualización de los pesos de las conexiones W_{ij} como

$$\Delta W_{ij} = -\eta \frac{\partial E}{\partial W_{ij}} =$$
$$\Delta W_{ij} = \eta \sum_{\mu}(\zeta_i^{\mu} - O_i^{\mu}) g'(h_i^{\mu}) V_j^{\mu} =$$
$$\Delta W_{ij} = \eta \sum_{\mu} \delta_i^{\mu} V_j^{\mu}$$

donde δ_i^{μ} queda definida como $\delta_i^{\mu} = (\zeta_i^{\mu} - O_i^{\mu}) g'(h_i^{\mu})$.

Redes neuronales artificiales

Perceptron multicapa

- Fijarse que la ecuación de ΔW_{ij} es muy similar a la del perceptron simple, pero en este caso V juega el rol que ocupaba ξ.

Perceptron multicapa

- En el caso Δw_{jk} usamos la $\frac{\partial E}{\partial w_{jk}}$

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}} =$$

$$\Delta w_{jk} = -\eta \sum_{\mu} \frac{\partial E}{\partial V_j^{\mu}} \frac{\partial V_j^{\mu}}{\partial w_{jk}} =$$

$$\Delta w_{jk} = \eta \sum_{\mu i} (\zeta_i^{\mu} - O_i^{\mu}) g'(h_i^{\mu}) W_{ij} g'(h_j^{\mu}) \xi_k^{\mu}$$

Perceptron multicapa

- Como ya habíamos definido
$$\delta_i^\mu = (\zeta_i^\mu - O_i^\mu) g'(h_i^\mu),$$
podemos re-escribir Δw_{jk} como
$$\Delta w_{jk} = \eta \sum_{\mu i} \delta_i^\mu W_{ij} g'(h_j^\mu) \xi_k^\mu.$$

Perceptron multicapa

- Así como para escribir ΔW_i definimos δ_i, podemos hacer lo mismo con Δw_{jk}.
  De este modo
  $$\Delta w_{jk} = \eta \sum_{\mu} \delta_j^{\mu} \xi_k^{\mu},$$
  donde
  $$\delta_j^{\mu} = g'(h_j^{\mu}) \sum_i W_{ij} \delta_i^{\mu}.$$

Perceptron multicapa

- Si prestamos especial atención a esta ecuación

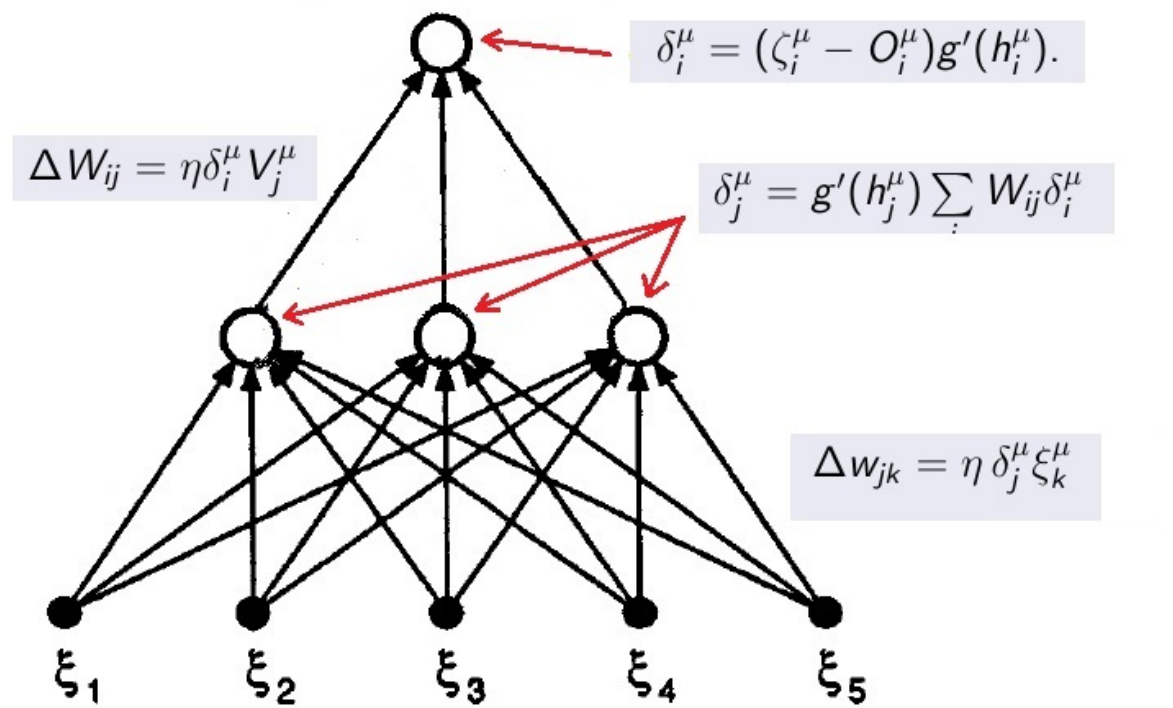
$$\delta_j^\mu = g'(h_j^\mu) \sum_i W_{ij} \delta_i^\mu$$

observamos que permite propagar δ_i^μ (el error en el capa de salida) a δ_j^μ (el error en la capa oculta) mediante los pesos W_{ij} .

- Es decir, esta ecuación permite propagar el error hacia atrás y de ahí su nombre de 'retropropagación' (backpropagation).

Redes neuronales artificiales

Perceptron multicapa



Introducción - Aprendizaje en el perceptron

Si, en vez de una capa oculta, tuvieramos más de una capa entre la entrada y la salida, la ecuación general para la actualización de pesos entre dos capas consecutivas $m-1$ y m, donde las unidades en la capa $m-1$ están indexadas por j y en la capa m por i, es :

$$\Delta W_{ij} = \eta \sum_{\mu} \delta_i^{\mu} V_j^{\mu}$$

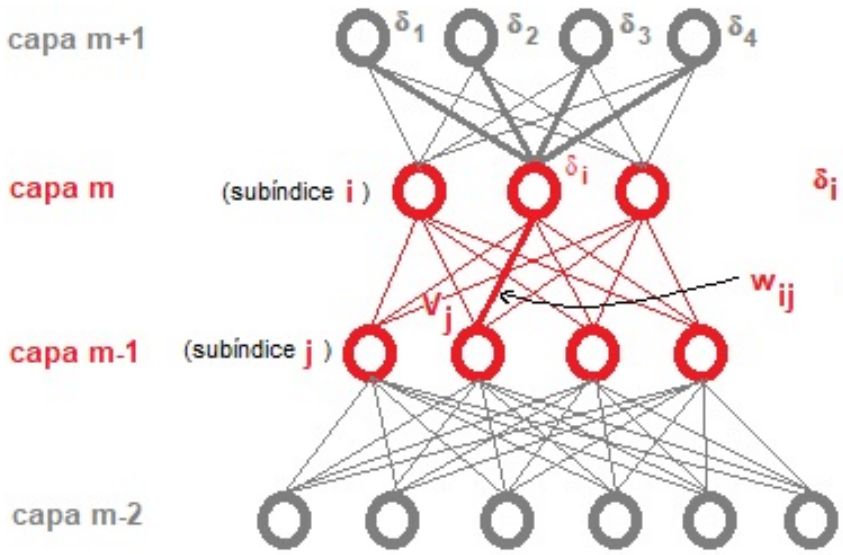
donde V_j^{μ} son el estado de activación de las unidades en la capa $m-1$.

Para una entrada μ dada, la ecuación es más simple aún:

$$\Delta W_{ij} = \eta \delta_i^{\mu} V_j^{\mu}$$

Redes neuronales artificiales

Perceptron multicapa



$$\delta_i = \delta_1 w_{1i} + \delta_2 w_{2i} + \delta_3 w_{3i} + \delta_4 w_{4i}$$

$$\Delta w_{ij} = \eta \, \delta_i V_j$$

Redes neuronales artificiales

Aprendizaje en el perceptron

- Fijarse que, la actualización de los pesos entre la capa $m-1$ y m (donde $m-1$ y m son capas consecutivas) depende de δ_i^μ el cual es propio para cada unidad de la capa m y de V_j^μ el cual es propio de cada unidad de la capa $m-1$.
- Esta característica es muy apropiada para calcular en forma paralela todos los pesos de las conexiones de la red (una vez que sea hayan retropropagado el error desde la salida hasta la capa siguiente de la entrada).

Redes neuronales artificiales

Secuencia del aprendizaje

La actualización de los pesos de las conexiones en una red puede realizarse de dos formas:

- incremental, ó
- en lotes.

Redes neuronales artificiales

Aprendizaje incremental

- En la forma incremental, cada vez que una entrada es presentada a la red, se propaga dicha entrada hasta obtener los estados de activación de la capa de salida.
- Luego, se retropropaga el error obteniendo los δ para cada unidad de cada capa.
- Finalmente se actualiza los pesos de todas las conexiones.
- Esto se repite para cada entrada μ.

Redes neuronales artificiales

Aprendizaje incremental

- Una vez que se han presentado todas las entradas del conjunto de entrenamiento, se ha completado una 'época'.
- El orden con que se presentan las entradas a la red se recomienda que sea al azar.

Redes neuronales artificiales

Aprendizaje en lote

- En la forma en lote, cada vez que una entrada es presentada a la red, se propaga dicha entrada hasta obtener los estados de activación de la capa de salida.
- Luego, se retropropaga el error obteniendo los δ para cada unidad de cada capa.
- Finalmente se acumula, para cada conexión, el Δ correspondiente.
- Esto se repite para cada entrada μ.

Redes neuronales artificiales

Aprendizaje en lote

- Una vez que se han presentado todas las entradas del conjunto de entrenamiento, se ha completado una 'época'.
- En ese momento se actualizan los pesos de las conexiones sumando el acumulado obtenido durante la presentación del lote.
- El orden con que se presentan las entradas a la red no altera el cálculo de los acumulados.

Redes neuronales artificiales

Algoritmo. Notación.

- Consideremos una red con varias capas ocultas cuyo índice de capa es m y varía entre 0 y M.
- La capa 0 es la capa de entrada.
- La capa M es la capa de salida.
- El *error* será calculado entre el valor deseado ζ_i^μ y el estado de activación de la unidad i de salida cuando el ejemplo ξ^μ es presentado en la capa 0 de la red (entrada).
- El aprendizaje será incremental.

Redes neuronales artificiales

Algoritmo. Notación.

- El estado de activación de cada unidad i en cada capa m será

$$V_i^m = g(\sum_j w_{ij}^m V_j^{m-1})$$

donde w_{ij}^m son los pesos de las conexiones entre la capa $m-1$ y m.

- El δ en cada unidad i de la capa de salida será

$$\delta_i^M = g'(h_i^M)(\zeta_i^\mu - V_i^M)$$

- El δ en cada unidad i de la capa $m-1$ será

$$\delta_i^{m-1} = g'(h_i^{m-1}) \sum_j w_{ji}^m \delta_j^m.$$

Redes neuronales artificiales

Algoritmo

1.Inicializar el conjunto de pesos en valores 'pequeños' al azar.
2.Tomar un ejemplo ξ^μ al azar del conjunto de entrenamiento y aplicarlo a la capa 0:
$V_k^0 = \xi_k^\mu$ para todo k.
3.Propagar la entrada hasta a capa de salida
$V_i^m = g(h_i^m) = g(\sum_j w_{ij}^m V_j^{m-1})$ para todo m desde 1 hasta M.
4.Calcular δ para la capa de salida
$\delta_i^M = g'(h_i^M)(\zeta_i^\mu - V_i^M)$
5.Retropropagar δ_i^M
$\delta_i^{m-1} = g'(h_i^{m-1}) \sum_j w_{ji}^m \delta_j^m$ para todo m entre M y 2
6.Actualizar los pesos de las conexiones de acuerdo
$w.nuevo_{ij}^m = w.viejo_{ij}^m + \Delta w_{ij}^m$
donde $\Delta w_{ij}^m = \eta \delta_i^m V_j^{m-1}$
7.Calcular el *error*. Si $error > COTA$, ir a 2.

Redes neuronales artificiales

Consideraciones acerca del método de retropropagación.

¿Qué tenemos hasta ahora?

- Olvidemonos por un instante de las neuronas, los pesos sinápticos, etc y concentremos nuestra atención en el método.
- Tenemos un conjunto de ejemplos y cada uno de ellos tiene una salida asociada (salida deseada).
- Con esos datos pretendemos aprender la relación que existe entre los ejemplos y sus salidas asociadas.
- Una vez que la hayamos aprendido, para cada entrada que le presentemos, nos dirá la salida que debería tener.

Redes neuronales artificiales

Consideraciones acerca del método de retropropagación.

- Y esto, no sólo cuando presentemos entradas que son parte del conjunto de ejemplos de entrenamiento, sino que, en principio, para cualquier entrada nueva que se nos ocurra.
- La dependencia entre las entradas y salidas puede ser no lineal, no necesita que las salidas(*) ni las entradas estén en un rango de valores, no hay limitación en la dimensión de la entrada ni de la salida.

  (*)dependiendo de la función de activación de las unidades de la capa de salida.

Redes neuronales artificiales

Consideraciones acerca del método de retropropagación.

- ¡Maravilloso!

Redes neuronales artificiales

Consideraciones acerca del método de retropropagación.

- Para obtener la red necesitamos "sólo"saber los pesos de las conexiones entre la neuronas.
- Pero, dichos pesos están en los reales, de modo que el espacio de posibles valores de los pesos es infinito.
- Es decir, es como buscar un átomo en el universo.

Redes neuronales artificiales

Consideraciones acerca del método de retropropagación.

- Sin embargo, la regla de actualización basada en la técnica del gradiente nos presta una invaluable ayuda.
- Nos permite recorrer ese espacio de valores de los pesos de una manera guiada, marcando un camino que nos conduzca al valor de la solución.

Redes neuronales artificiales

Consideraciones acerca del método de retropropagación.

- Si fueran 2 los pesos de las conexiones que tenemos que hallar, nosotros podríamos representar a la función E(w) en un gráfico 3D.
- Veríamos algo similar a lo ocurre en una cadena montañosa, con picos y valles y lo que estamos buscando es el valle más profundo donde el valor de la función de costo E(w) será mínimo.

Redes neuronales artificiales

Consideraciones acerca del método de retropropagación.

- El método del gradiente descendente nos da una dirección en la cual lograremos navegar esa superficie bajando cada vez el costo.
- Esa dirección es ΔW.

Redes neuronales artificiales

Consideraciones acerca del método de retropropagación.

- El problema que tiene el método del gradiente descendente es que la información que dispone es local.
- Esto es, porque la dirección de descenso es, justamente, la derivada de E(W) respecto a cada uno de los pesos de las conexiones en el punto obtenido hasta ese momento.

Consideraciones acerca del método de retropropagación.

- Y eso es lo que da como resultado que el método nos conduce a mínimos locales.
- Y por la propia naturaleza del método, cuando se llega a un mínimo local no hay dirección posible que nos conduzca a seguir bajando.

Redes neuronales artificiales

Consideraciones acerca del método de retropropagación.

- ¡Cáspitas! Era muy 'maravilloso' el método...

Consideraciones acerca del método de retropropagación.

- Un problema que suele aparecer es el siguiente.
- El error que se retropropaga depende de $(\zeta_i^\mu - O_i^\mu)$.
- Cuando el algoritmo se acerca al mínimo global de $E()$, esta diferencia se acerca a 0.
- Esto produce que el ΔW que depende del error retropropagado se hace muy pequeño y el descenso de $E()$ entra en una zona plana sin casi modificarse a lo largo del tiempo.

Consideraciones acerca del método de retropropagación.

- Una alternativa es definir una función de costo que evite esto.
- Definamos $E()$ como

$$E(w) = \sum_{i\mu}(\tfrac{1}{2}(1+\zeta_i^\mu) log \tfrac{1+\zeta_i^\mu}{1+O_i^\mu} + \tfrac{1}{2}(1-\zeta_i^\mu) log \tfrac{1-\zeta_i^\mu}{1-O_i^\mu})$$

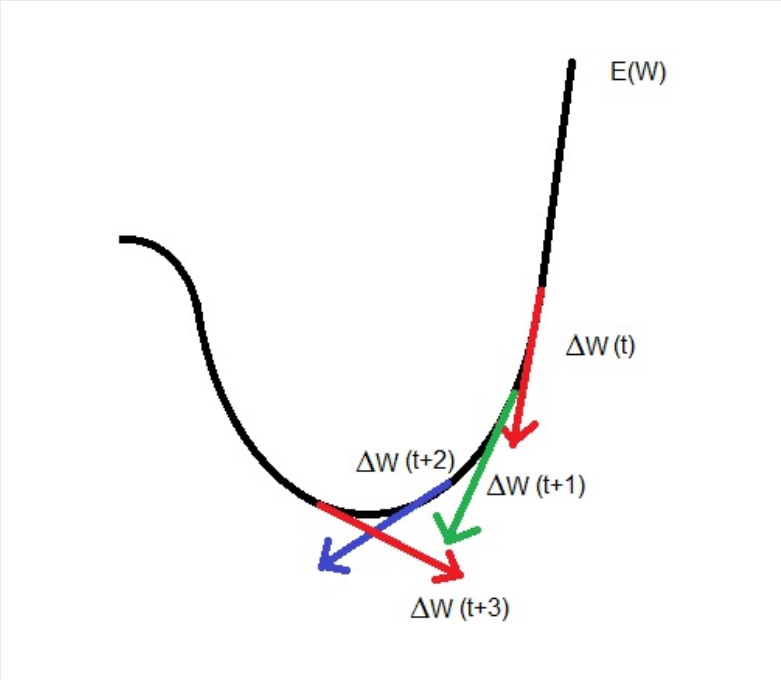
- Fijarse que cuando O_i^μ se acerca a ζ_i^μ, entonces la función $E()$ tiende a 0 y cuando se aleja, tiende a ∞.

Consideraciones acerca del método de retropropagación.

- Si se usa como función de activación la *tanh*() y $\zeta_i^\mu \in \{1, -1\}, O_i^\mu$ se acercará mucho a ζ_i^μ pero nunca adoptará ese valor.
- Por otro lado, al diferenciar la función $E()$, el δ_i^μ quedará defindo como
$$\delta_i^\mu = \beta(\zeta_i^\mu - O_i^\mu)$$
donde β corresponde a la definición de $g(h) = tanh(\beta h)$.
- Fijarse que desapareció el factor $g'(h_i^\mu)$ del cálculo de δ_i^μ.

Momentum

- Otro inconveniente que suele pasar cuando el W está cerca de un mínimo (sea local o global) es que puede quedar oscilando en torno al mínimo e incluso diverger.



Momentum

- Una forma de abordar este problema es el momentum:
$$\Delta w_{ij}(t+1) = -\eta \frac{\partial E}{\partial w_{ij}} + \alpha \Delta w_{ij}(t)$$
donde α está entre 0 y 1,
- $\alpha = 0{,}8$ o $\alpha = 0{,}9$ son frecuentemente usados.
- Si la función $E()$ está en una región plana, $\Delta w_{ij}(t)$ será similar a $\frac{\partial E}{\partial w_{ij}}$ y por lo tanto $\Delta w_{ij}(t+1)$ será verá incrementado en módulo.
- Si la función $E()$ está en un valle o una región oscilatoria, $\Delta w_{ij}(t+1)$ tenderá a compensarse para evitar oscilaciones.

η adaptativo

- Si, en forma consistente, $E()$ decrece en un momento del aprendizaje (por ejemplo, durante k iteraciones seguidas), quizás sería interesante incrementar a η.
- Por otro lado, si $E()$ comienza a incrementarse en un momento del aprendizaje (por ejemplo, durante k' iteraciones seguidas) quizás sería interesante decrementar a η para evitar divergencia.

η adaptativo

- Esto se puede formalizar adaptando el valor de η según

$$\Delta\eta = \begin{cases} +a & \text{si } \Delta E < 0 \text{ en forma consistente} \\ -b\eta & \text{si } \Delta E > 0 \text{ empieza a ser consistente} \\ 0 & \text{en otro caso} \end{cases}$$

η adaptativo

- También es posible actualizar el η por neurona (para los pesos de las conexiones que llegan a esa neurona) o por conexión (un η por cada w_{ij}).

Redes neuronales artificiales

Otros métodos de minimización

- Estas modificaciones fueron propuestas basadas en el algoritmo original donde la dirección de descenso (ΔW) está dada por el método del gradiente descendente.
- Sin embargo existen otras alternativas para encontrar la dirección de descenso las cuales no dependen exclusivamente del gradiente.
- Incluso, hay alternativas que no requieren que la función de costo sea derivable (o conocer la derivada de la función de costo).