¿Porqué es difícil entrenar redes profundas?

Ahora tenemos una idea bastante acabada de como funciona una red neuronal de un número arbitrario de capas con neuronas lineales o no lineales.

La esencia del método, como vimos, incluye

- Disponer de un conjunto de datos previamente etiquetados con la respuesta correcta. Este conjunto de datos nos permitirá entrenar nuestra red.
- reservar al menos una parte de este conjunto de datos etiquetados que usamos para evaluar como resuelve la red los casos que nunca ha visto. Solo si hace bien este paso, diremos que la red ha aprendido y es capaz de generalizar.
- Debemos definir una función error $E(\{\vec{w}\})$ que depende de todos los acoplamientos sinápticos de nuestra red. Por ahora usamos un error cuadrático.

- Preparamos la red eligiendo aleatoriamente todos los acoplamientos sinápticos.

- Ahora aplicamos el descenso por el gradiente determinista. Le presentamos secuencialmente los inputs del conjunto de entrenamiento y obtenemos los resultados O_i^M $(i=1,2,\dots,M)$. Con estos calculamos E, con E calculamos los incrementos de cada acoplamiento Δw_{pq}

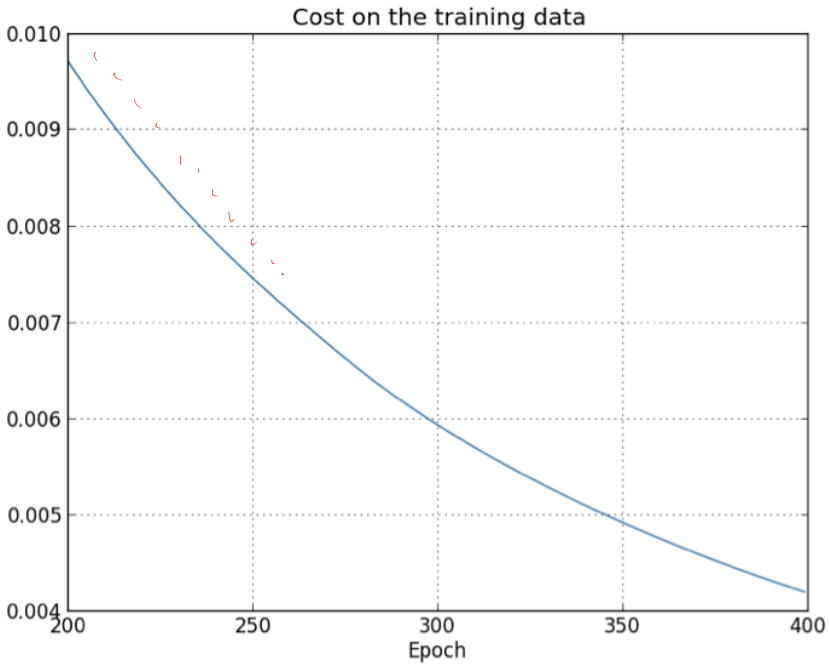
$$\Delta w_{pq} = -\eta \frac{\partial E}{\partial w_{pq}}$$

Cuando le hemos presentado los p patrones del conjunto de entrenamiento, hemos pasado 1 ÉPOCA. Calculamos el valor E en la época 1.

- Repetimos el último paso incrementando la época en 1

$$\text{época} =: \text{época} + 1$$

y monitoreamos el valor de E en función de época. Sabemos que este valor debe decrecer o mantenerse constante.



Ejemplo: E debe decrecer o mantenerse constante

Nos detenemos cuando el error alcanza cierta tolerancia preestablecida o cuando alcanzamos cierto número de épocas.

Ahora hacemos una evaluación de la capacidad de aprendizaje, época por época o al final.

Se puede mostrar que con una capa oculta con un número arbitrario de neuronas es posible aproximar tan bien como queramos una función dada, bajo condiciones específicas. Pero esto requiere un número exponencialmente grande de neuronas cuando aumenta la complejidad del problema.

En una red de una capa oculta vimos que la salida O_i depende de la entrada a través de las neuronas de la capa intermedia V

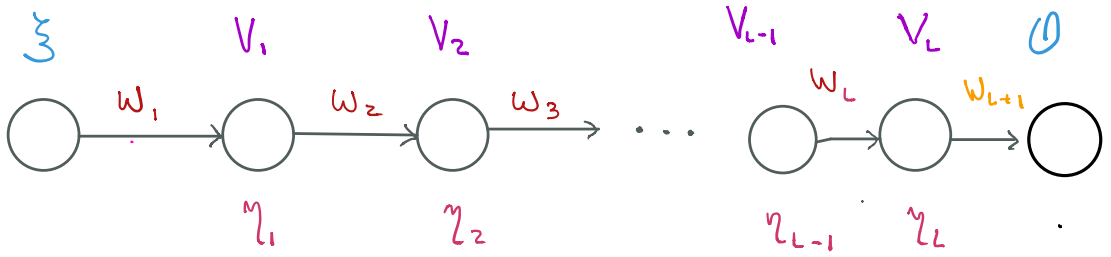
$$O_i = g\left(\sum_j W_{ij}\; g\left(\sum_{k=1} W_{jk}\,\xi_k\right)\right)$$

Incluso podemos tener funciones diferentes en la capa intermedia y en la de salida:

$$O_i^\mu = g_1\left(\sum_j W_{ij}\cdot \overbrace{g_2\left(\underbrace{\sum_{k=1} W_{jk}\,\xi_k^\mu}_{h_j}\right)}^{V_j(h_j)}\right)$$

Con estos O_i^μ calculamos los δ^μ y con ellos "volvemos" de la salida a la entrada cambiando los acoplamientos.

La red más simple, y por cierto inútil, es una red de muchas capas con una única neurona por capa



$$O = g_{L+1}(w_{L+1}V_L - \eta_L) = g_{L+1}(w_{L+1}\, g_L(w_L V_{L-1} - \eta_{L-1}) - \eta_L)$$

$$= g_{L+1}(w_{L+1}\, g_L(w_L\, g_{L-1}(w_{L-1}V_{L-2} - \eta_{L-2}) - \eta_{L-1}) - \eta_L)$$

$$= g_{L+1}(w_{L+1}\, g_L(w_L\, g_{L-1}(w_{L-1}\, g_{L-2}(w_{L-2}\, g_{L-3}(w_{L-3}\, g_{L-4}\cdots)\cdots - \eta_L)$$

Hacia adentro componemos funciones

$$\frac{\partial E}{\partial w_1} = g_1'(h_1) \times w_2 \times g_2'(h_2) \times \cdots \times w_L\, g_L'(h_L) \times \underbrace{[\delta^\mu - O^\mu]}_{\text{diferencia entre deseado y real}}$$

Hacia atrás multiplicamos derivadas de funciones

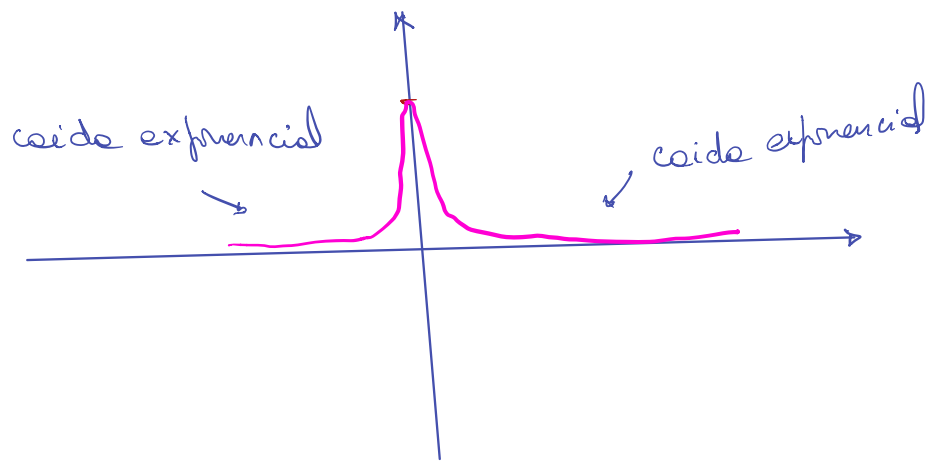
Veamos en seguida las dos funciones que se usaron originalmente:

$$g(h) = \tanh(h) = \frac{e^{h} - e^{-h}}{e^{h} + e^{-h}} = \frac{\operatorname{senh}(h)}{\cosh(h)}$$

$$g'(h) = \frac{(e^{h}+e^{-h})(e^{h}+e^{-h}) - (e^{h}-e^{-h})(e^{h}-e^{-h})}{(e^{h}+e^{-h})^{2}}$$

$$= \frac{(e^{h}+e^{-h})^{2} - (e^{h}-e^{-h})^{2}}{(e^{h}+e^{-h})^{2}}$$

$$= 1 - \tanh^{2}(h) = (1 - g^{2}(h))$$



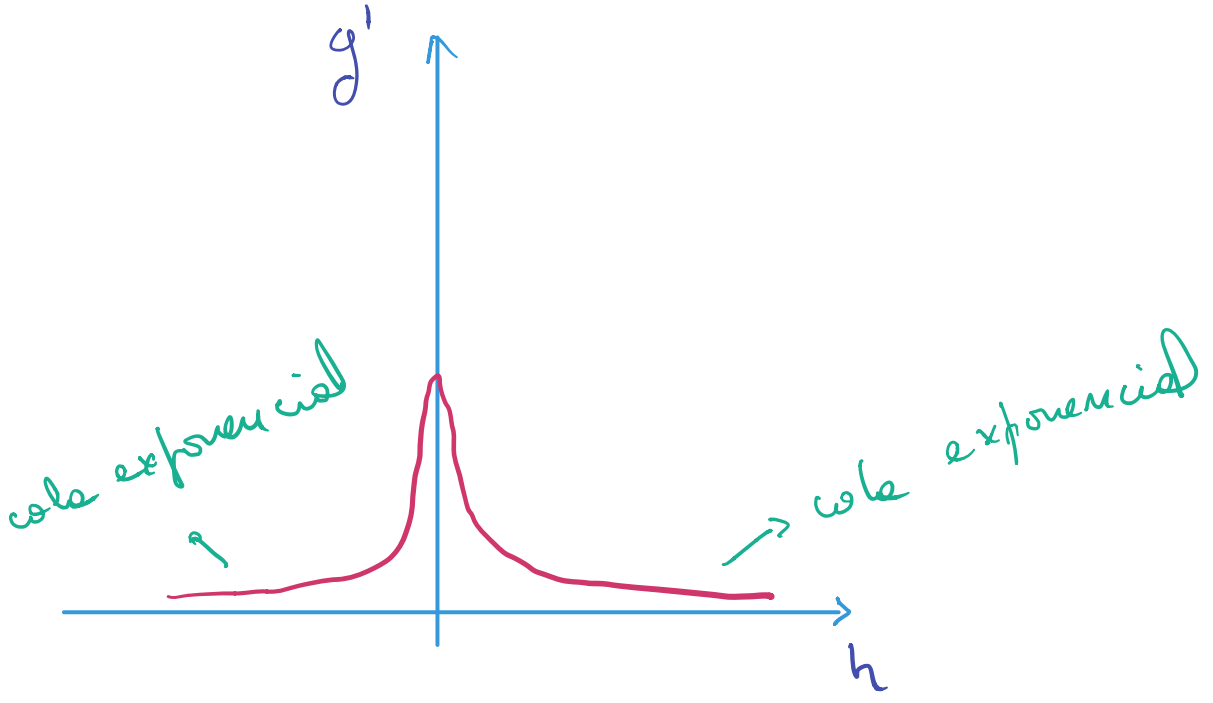
Para neuronas sigmoidales:

$$g(h) = \frac{1}{1+e^{-h}}$$

$$g'(h) = \frac{-(-e^{-h})}{(1+e^{-h})^2} = \frac{e^{-h}}{(1+e^{-h})^2}$$

$$= \frac{1}{(1+e^{-h})} \frac{e^{-h}}{(1+e^{-h})} = g(h) \cdot \left(1 - \frac{1}{1+e^{-h}}\right)$$

$$= g(h)\,(1 - g(h))$$



Noten que $g(4)$ tiene una imagen exponencialmente pequeña salvo en un entorno de cero con un ancho típico de uno. Entonces cuando tenemos muchas neuronas en muchas capas, a medida que avanza el back-propagation hacia atrás, los incrementos se hacen exponencialmente pequeños y la red no puede aprender. Esto se llama

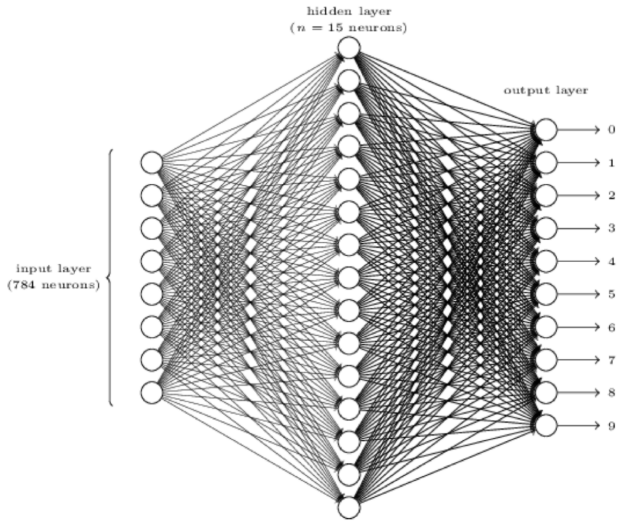
SUPRECIÓN DEL GRADIENTE

Miremos un ejemplo.

Se hizo una red multicapas para aprender a distinguir dígitos escritos a mano sacados de la base de datos MNIST.

La entrada son 724 neuronas correspondientes a los 28×28 pixeles de las fotos de MNIST pasados de matriz bidimensional a vector (linear).

Hacemos redes feed-forward cada vez más profundas para aprender a reconocer los 10 dígitos (MNIST).



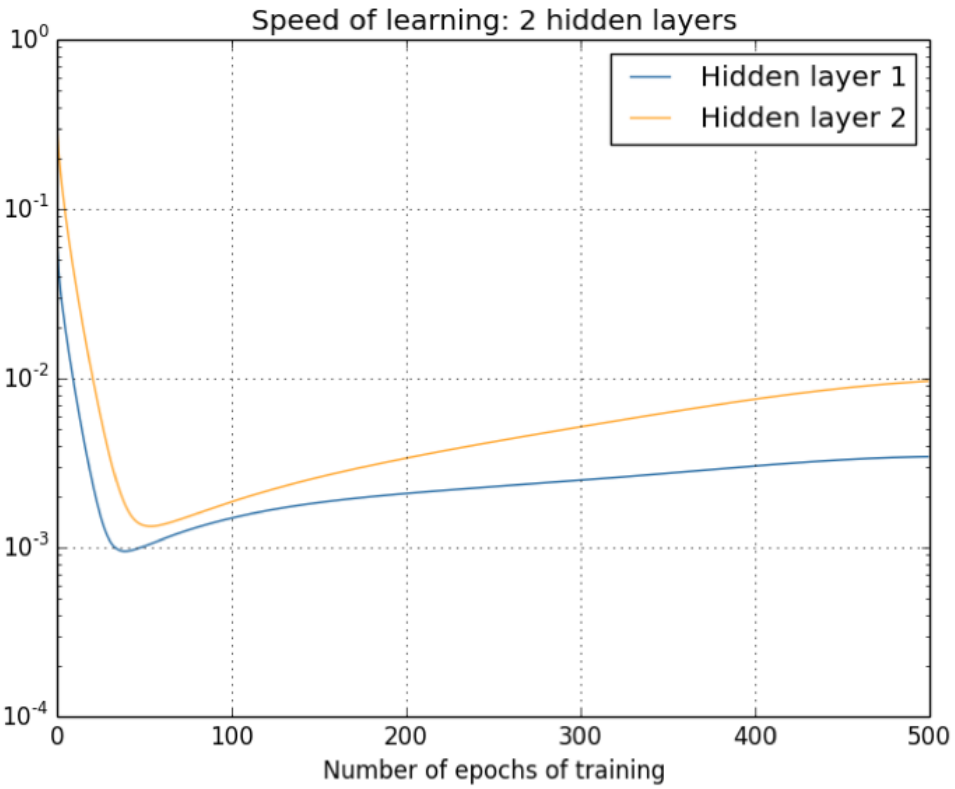
Miremos qué sucede con un acoplamiento en particular a medida que hacemos más profunda la red, comenzando con 2 capas ocultas.

Miraremos como cambia la rapidez de cambio del umbral de la primera neurona de la primera capa oculta ($\eta = w_0^{(1)}$)

Esta velocidad se representa con los cuadros azules

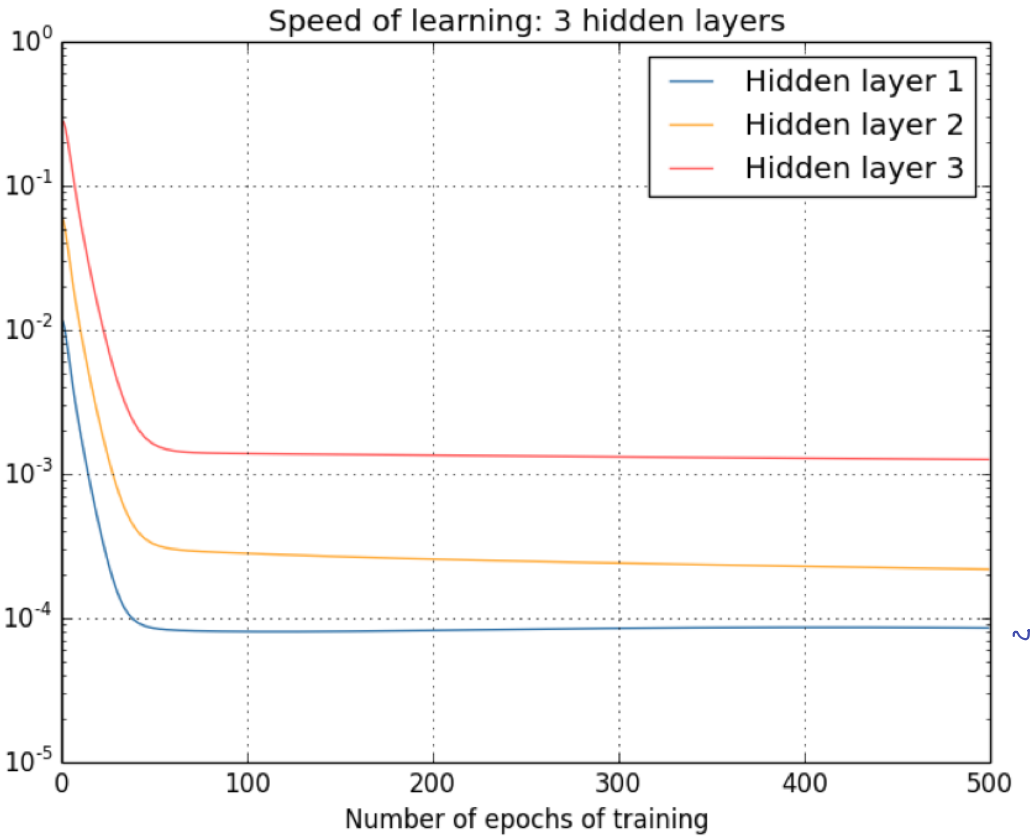
Cada capa oculta tiene 100 neuronas.

2 CAPAS OCULTAS



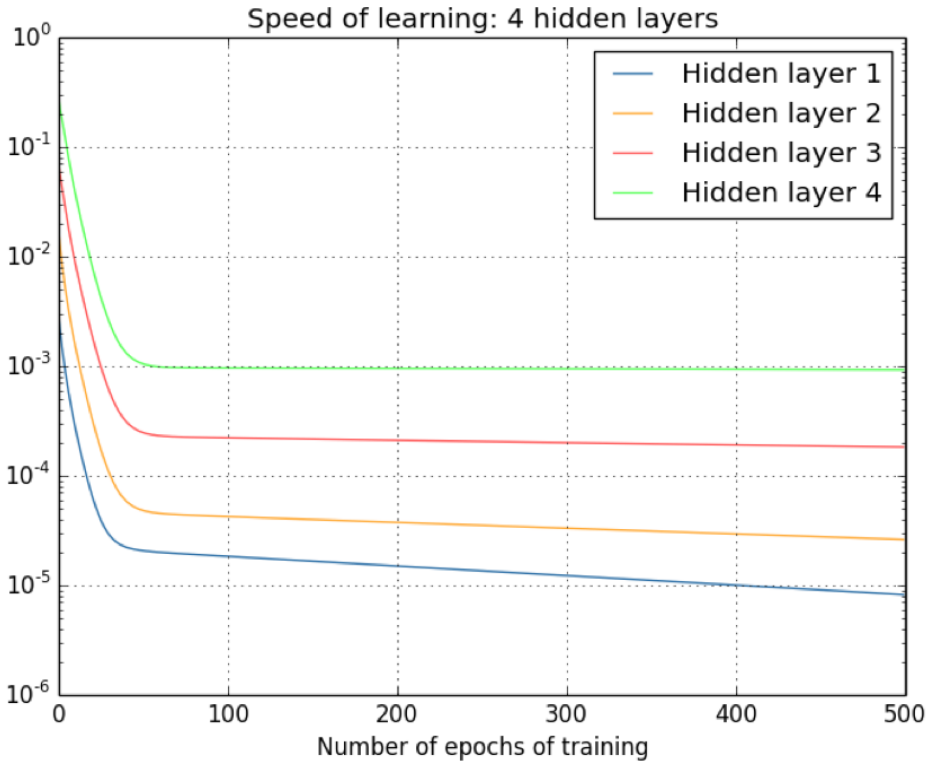
$\sim 2{,}6 \times 10^{-2}$

3 CAPAS OCULTAS



$\sim 9 \times 10^{-5}$

4 CAPAS OCULTAS



$\sim 9 \times 10^{-6}$

Vemos que cuanto más capas ponemos, la última capa de sinapsis aprende siempre a la misma velocidad (aproximadamente) y la primera cada vez más lentamente.

SUPRESIÓN DEL GRADIENTE

Esto es culpa de varias cosas:

* las funciones sigmoidal y tangente tienen derivadas pequeñas, y cada vez que vamos hacia atrás, la razón de cambio se hace multiplicando más derivadas.

* el método de gradiente puede pasar en mínimos locales

* la razón de aprendizaje es fija.

* los W iniciales son aleatorios en un hiperespacio de muchas dimensiones

En la próxima clase veremos como solucionar estos problemas.