

Redes Neuronales Artificiales

Hernán Felipe García Arias

Machine Learning

UTP

Introducción

$$x_n \rightarrow \underline{\phi}_n \quad M \times 1$$

- Se han considerado modelos de regresión y clasificación que comprenden combinaciones lineales de funciones base fijas.
- Problemas de gran escala \rightarrow adaptar los parámetros a los datos.
- Adaptar los parámetros durante el entrenamiento.
- El modelo más exitoso de este tipo en el contexto del aprendizaje de máquina es el *perceptrón multicapa*.

Contenido

Redes de propagación hacia adelante



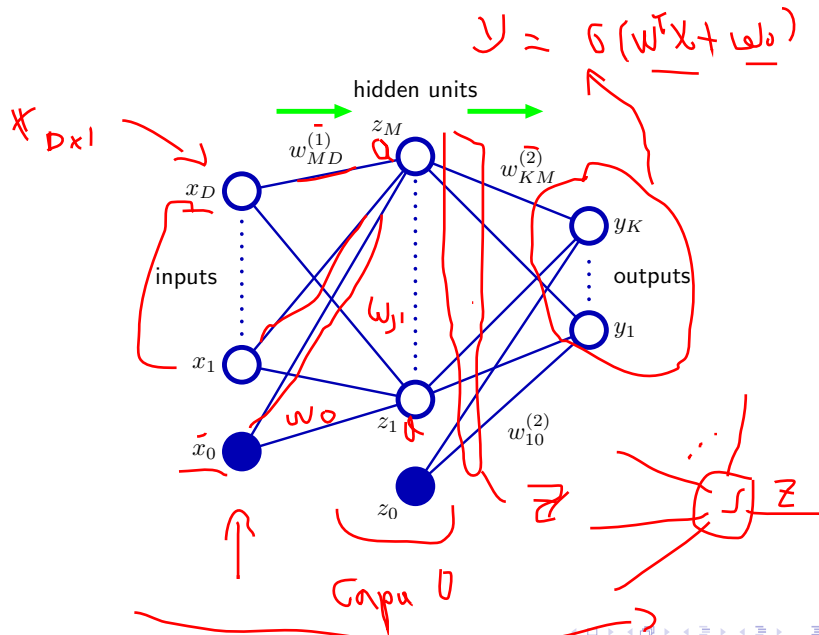
Historia de las redes neuronales

Entrenamiento de redes

Propagación del error hacia atrás

Regularización en redes neuronales

Arquitectura típica de una red neuronal



Modelo (I)

- El modelo básico de red neuronal puede describirse por una serie de transformaciones funcionales.
- Primero se construyen M combinaciones lineales de las variables de entrada x_1, \dots, x_D en la forma

$$a_j = \sum_{i=1}^D w_{j,i}^{(1)} x_i + w_{j,0}^{(1)},$$

donde $j = 1, \dots, M$, y el superíndice (1) indica los parámetros correspondientes a la primera “capa” de la red.

Modelo (II)

- Las cantidades a_j se conocen como *activaciones*.
- Las a_j se transforman usando una función de activación no lineal $h(\cdot)$ para dar

$$z_j = h(a_j).$$

- En este contexto, estas funciones se conoce como *nodos ocultos*.
- En los modelos lineales, las funciones base $\phi_j(\mathbf{x})$ equivalen a z_j .

Modelo (III)

- Los z_j se combinan de nuevo linealmente para dar *activaciones de salida*

$$\underline{a_k} = \sum_{j=1}^M \underline{w_{k,j}^{(2)}} \underline{z_j} + \underline{w_{k,0}^{(2)}}$$

donde $k = 1, \dots, K$, y K es el número total de salidas.

- El superíndice (2) indica los parámetros correspondientes a la segunda “capa” de la red.
- Las activaciones de salida se transforman o no, dependiendo del problema a tratar
 - Regresión $\rightarrow y_k = a_k$.
 - Clasificación $\rightarrow y_k = \sigma(a_k)$.

Modelo (IV)

- Combinando estas etapas se tiene

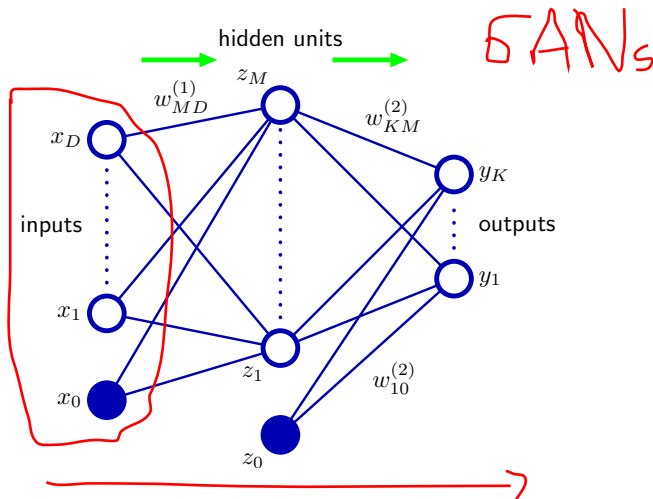
$$y_k(\mathbf{x}, \mathbf{w}) = \sigma \left(\sum_{j=1}^M w_{k,j}^{(2)} h \left(\sum_{i=1}^D w_{j,i}^{(1)} x_i + w_{j,0}^{(1)} \right) + w_{k,0}^{(2)} \right),$$
$$= \sigma \left(\sum_{j=0}^M w_{k,j}^{(2)} h \left(\sum_{i=0}^D w_{j,i}^{(1)} x_i \right) \right),$$

con $x_0 = 1$.

- Nótese que en la segunda igualdad, de la ecuación anterior se ha hecho $z_0 = h \left(\sum_{i=0}^D w_{0,i}^{(1)} x_i \right)$.
- Los parámetros $\{w_{j,i}\}_{j=1,i=0}^{M,D}$ y $\{w_{k,j}\}_{k=1,j=0}^{K,M}$ se denotan conjuntamente como \mathbf{w} , y se organizan como vector.
- Red neuronal: función no lineal de $\{x_i\}_{i=1}^D$ a $\{y_k\}_{k=1}^K$ controlada por \mathbf{w} .

Modelo (V)

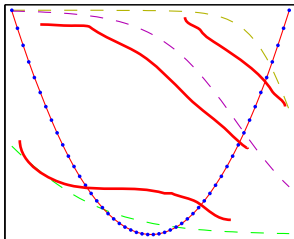
La red de la figura es una red de dos capas, debido a que dos es el número de capas con pesos adaptativos.



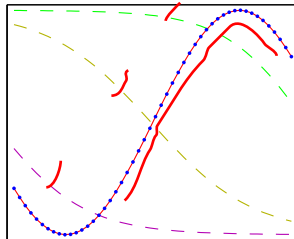
Propiedades de aproximación (I)

- Aproximadores universales.
- El problema principal consiste en encontrar valores de parámetros **w** adecuados dado un conjunto de datos de entrenamiento.

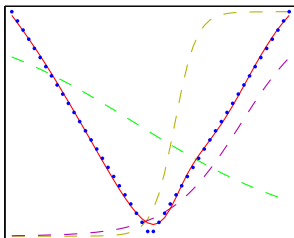
Propiedades de aproximación (II)



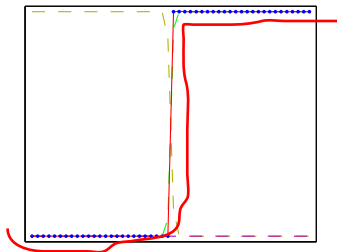
$$f(x) = x^2$$



$$f(x) = \sin(x)$$



$$f(x) = |x|$$



$$f(x) = u(x)$$

Contenido

Redes de propagación hacia adelante

Historia de las redes neuronales

Entrenamiento de redes

Propagación del error hacia atrás

Regularización en redes neuronales

Modelo de una neurona



- Se considera que las redes neuronales comenzaron con McCulloch y Pitts, en 1943.
- McCulloch y Pitts formularon un modelo matemático simple de una neurona.
- Aproximaron la salida que se quiere modelar como una suma ponderada de las entradas que pasan a través de una función de umbralización

$$y = \mathbb{I} \left(\underbrace{\sum_{\forall i} w_i x_i}_h > \theta \right),$$

para algún umbral θ .

Perceptrón y Adaline

$$W^* = W - \alpha \nabla E(W)$$

- Frank Rosenblatt inventó el algoritmo del perceptrón en 1957, que era una forma de estimar los parámetros de una neurona McCulloch y Pitts.
- In 1960, Widrow and Hoff inventaron un modelo muy similar conocido como **adaline** (adaptive linear element).
- En 1969, Minsky y Papert publicaron un libro conocido como “Perceptrones”, en el que mostraron que un modelo sin capas ocultas era muy limitado, porque no permitía clasificar datos que no fueran linealmente separables.
- Lo anterior redujo considerablemente el interés en el tema.

Backpropagation


- En 1986, Rumelhart, Hinton y Williams descubrieron el algoritmo de propagación hacia atrás, que permitía ajustar modelos con capas ocultas.
 - El algoritmo backpropagation fue descubierto originalmente por Bryston and Ho, en 1969.
 - Fue descubierto de forma independiente por Werbos en 1974.
 - Rumelhart y sus colaboradores llamaron la atención de la comunidad sobre la existencia del algoritmo.
- El descubrimiento del algoritmo de Backpropagation popularizó estos modelos por más de una década.

NETtalk y LeNet

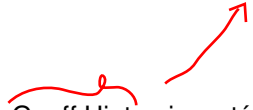
- En 1987, Sejnowski y Rosenberg crearon el sistema NETTalk, que aprendía a mapear palabras escritas en inglés a símbolos fonéticos que podían ser usados para alimentar un sintetizador de voz.
- En 1989, Yann Le Cun y otros crearon el sistema LeNet, que permitía hacer clasificación de dígitos manuscritos a partir de imágenes.

Máquinas de vectores de soporte



- ❑ Las máquinas de vectores de soporte fueron inventadas en 1992 (Boser et al. 1992).
 - ❑ Las SVMs tienen una capacidad de predicción similar a la de las redes neuronales, siendo mucho más fáciles de entrenar (porque la función objetivo es convexa).
- 
- ❑ Lo anterior inició una década de investigación sobre SVMs.
 - ❑ Las SVMs no usan funciones de base adaptativas, por lo cual requieren que un experto humano diseñe la función kernel adecuada.

Redes profundas (Deep networks)

- 
- ❑ En 2002, Geoff Hinton inventó una técnica de entrenamiento conocida como “contrastive divergence training procedure”.
 - ❑ La importancia de este algoritmo es que permitió, por primera vez, aprender redes profundas entrenando una capa a la vez, en una forma no supervisada.
 - ❑ Este descubrimiento ha generado de nuevo un interés en el área de las redes neuronales.

Contenido

Redes de propagación hacia adelante

Historia de las redes neuronales

Entrenamiento de redes

Propagación del error hacia atrás

Regularización en redes neuronales

Funciones de error: regresión

- **Regresión simple.** Dado un conjunto de entrenamiento $\{\mathbf{x}_n, t_n\}_{n=1}^N$ se minimiza una función de error

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \|y(\mathbf{w}, \mathbf{x}_n) - t_n\|^2.$$

- **Regresión múltiple.** Dado un conjunto de entrenamiento $\{\mathbf{x}_n, \mathbf{t}_n\}_{n=1}^N$ se minimiza una función de error

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \|\mathbf{y}(\mathbf{w}, \mathbf{x}_n) - \mathbf{t}_n\|^2.$$

Funciones de error: clasificación binaria

- Se tiene una única variable t tal que $t = 1$ denota la clase \mathcal{C}_1 y $t = 0$ denota la clase \mathcal{C}_2 .
- Se puede demostrar que la función de error a minimizar es

$$E(\mathbf{w}) = - \sum_{n=1}^N t_n \ln y_n + (1 - t_n) \ln(1 - y_n),$$

donde $y_n = \sigma(a_n) = 1/(1 + \exp(-a_n))$, y a_n es la activación de la capa de salida.

- Para el caso de K problemas binarios, la función a minimizar es

$$E(\mathbf{w}) = - \sum_{n=1}^N \sum_{k=1}^K t_{n,k} \ln y_{n,k} + (1 - t_{n,k}) \ln(1 - y_{n,k}),$$

donde $t_{n,k} \in \{0, 1\}$, y $y_{n,k} = \sigma(a_{n,k})$.

Funciones de error: clasificación de múltiples clases

- El vector \mathbf{t}_n tiene una codificación 1 de K .
- La función de error a minimizar es

$$E(\mathbf{w}) = - \sum_{n=1}^N \sum_{k=1}^K t_{n,k} \ln y_k(\mathbf{w}, \mathbf{x}_n),$$

donde

$$y_k(\mathbf{w}, \mathbf{x}_n) = \frac{\exp(a_k(\mathbf{w}, \mathbf{x}_n))}{\sum_j \exp(a_j(\mathbf{w}, \mathbf{x}_n))}.$$

Optimización de parámetros

- El objetivo es encontrar un vector \mathbf{w} que minimice la función $E(\mathbf{w})$.
- Optimización por gradiente descendente

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta E(\mathbf{w}),$$

donde $\eta > 0$ se conoce como la razón de aprendizaje.

- Otros métodos de optimización incluyen gradiente conjugado, o los métodos de cuasi-Newton.

Contenido

Redes de propagación hacia adelante

Historia de las redes neuronales

Entrenamiento de redes

Propagación del error hacia atrás

Regularización en redes neuronales

Funcionamiento

- ❑ El algoritmo de propagación del error hacia atrás (backpropagation) es una forma eficiente de evaluar el gradiente de la función de error $E(\mathbf{w})$.
- ❑ Se realiza en dos etapas
 - Primera etapa → evaluación de las derivadas de la función error con respecto a \mathbf{w} .
 - Segunda etapa → las derivadas se usan para realizar los ajustes de los pesos, usando un método de optimización.

Derivadas de la función de error (I)

- Asumiendo que los datos son iid, las funciones de error toman la forma

$$E(\mathbf{w}) = \sum_{n=1}^N E_n(\mathbf{w}),$$

donde $E_n(\mathbf{w})$ es el error calculado sobre cada dato.

- Recordemos cómo es la salida del modelo

$$y_k(\mathbf{x}_n, \mathbf{w}) = f \left(\sum_{j=0}^M w_{k,j}^{(2)} h \left(\sum_{i=0}^D w_{j,i}^{(1)} x_{n,i} \right) \right),$$

donde $f(\cdot)$ es lineal o sigmoidal dependiendo del problema a tratar.

- Para usar los algoritmos de optimización es necesario calcular el gradiente $\frac{dE(\mathbf{w})}{d\mathbf{w}}$.

Derivadas de la función de error (II)

- A manera de ilustración, supongamos que se trata de un problema de clasificación binaria. En este caso

$$E(\mathbf{w}) = \sum_{n=1}^N E_n(\mathbf{w}) = - \sum_{n=1}^N t_n \ln y_n + (1 - t_n) \ln(1 - y_n),$$

donde $y_n = \sigma(a_n)$, y a_n es la activación de la salida dada como

$$a_n = \sum_{j=0}^M w_{1,j}^{(2)} h \left(\sum_{i=0}^D w_{j,i}^{(1)} x_{n,i} \right),$$

Es decir,

$$y_n = \sigma \left(\sum_{j=0}^M w_{1,j}^{(2)} h \left(\sum_{i=0}^D w_{j,i}^{(1)} x_{n,i} \right) \right),$$

- En general, para problemas de K salidas se escribe $a_{n,k}$, $y_{n,k}$ y los pesos de la primera capa son $w_{k,j}^{(2)}$.

Derivadas de la función de error (III)

- Se define $\mathbf{w} = [\mathbf{w}_1^\top \mathbf{w}_2^\top]^\top$, donde \mathbf{w}_1 y \mathbf{w}_2 son los vectores de pesos

$$\mathbf{w}_1 = \begin{bmatrix} w_{1,0}^{(1)} & \cdots & w_{M,0}^{(1)} & w_{1,1}^{(1)} & \cdots & w_{M,1}^{(1)} & \cdots & w_{1,D}^{(1)} & \cdots & w_{M,D}^{(1)} \end{bmatrix}^\top$$
$$\mathbf{w}_2 = \begin{bmatrix} w_{1,0}^{(2)} & \cdots & w_{K,0}^{(2)} & w_{1,1}^{(2)} & \cdots & w_{K,1}^{(2)} & \cdots & w_{1,M}^{(2)} & \cdots & w_{K,M}^{(2)} \end{bmatrix}^\top.$$

- Es necesario encontrar

$$\frac{dE(\mathbf{w})}{d\mathbf{w}} = \left[\left(\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}_1} \right)^\top \quad \left(\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}_2} \right)^\top \right]^\top,$$

donde $\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}_1}$ tiene elementos $\frac{\partial E(\mathbf{w})}{\partial w_{j,i}^{(1)}}$, y $\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}_2}$ tiene elementos $\frac{\partial E(\mathbf{w})}{\partial w_{k,j}^{(2)}}$.

Derivadas de la función de error (IV)

- Nos enfocamos ahora en encontrar las derivadas $\frac{\partial E(\mathbf{w})}{\partial w_{j,i}^{(1)}}$, y $\frac{\partial E(\mathbf{w})}{\partial w_{k,j}^{(2)}}$.
- Regla de la cadena.
- La derivada de $E(\mathbf{w})$ con respecto a \mathbf{w} se calcula como

$$\nabla E(\mathbf{w}) = \sum_{n=1}^N \nabla E_n(\mathbf{w}).$$

Derivadas de la función de error (V)

- Se necesita encontrar

$$\frac{\partial E_n(\mathbf{w})}{\partial w_{1,j}^{(2)}} = \frac{\partial E_n(\mathbf{w})}{\partial a_{n,1}} \frac{\partial a_{n,1}}{\partial w_{1,j}^{(2)}}, \quad \frac{\partial E_n(\mathbf{w})}{\partial w_{k,j}^{(2)}} = \frac{\partial E_n(\mathbf{w})}{\partial a_{n,k}} \frac{\partial a_{n,k}}{\partial w_{k,j}^{(2)}}.$$

- Se tiene además

$$a_{n,1} = \sum_{j=0}^M w_{1,j}^{(2)} z_{n,j}, \quad a_{n,k} = \sum_{j=0}^M w_{k,j}^{(2)} z_{n,j}.$$

- Nótese entonces que

$$\frac{\partial a_{n,1}}{\partial w_{1,j}^{(2)}} = z_{n,j}, \quad \frac{\partial a_{n,k}}{\partial w_{k,j}^{(2)}} = z_{n,j}.$$

Derivadas de la función de error (VI)

P	$E_n(\mathbf{w})$	$y_n(a_n), y_{nk}(a_{nk})$	$\frac{\partial E_n(\mathbf{w})}{\partial a_n}, \frac{\partial E_n(\mathbf{w})}{\partial a_{nk}}$
RS	$\frac{1}{2}(t_n - y_n)^2$	$y_n = a_n$	$y_n - t_n$
RM	$\frac{1}{2} \sum_{k=1}^K (t_{n,k} - y_{n,k})^2$	$y_{n,k} = a_{n,k}$	$y_{n,k} - t_{n,k}$
CS	$-\{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\}$	$y_n = \sigma(a_n)$	$y_n - t_n$

P: Problema. RS: Regresión Simple. RM: Regresión Múltiple. CS: Clasificación Simple.

Derivadas de la función de error (VII)

- Se necesita encontrar

$$\frac{\partial E_n(\mathbf{w})}{\partial w_{j,i}^{(1)}} = \frac{\partial E_n(\mathbf{w})}{\partial a_{n,j}} \frac{\partial a_{n,j}}{\partial w_{j,i}^{(1)}}.$$

- Para la derivada más interna se tiene

$$a_{n,j} = \sum_{i=0}^D w_{j,i}^{(1)} x_{n,i}.$$

- Nótese entonces que

$$\frac{\partial a_{n,j}}{\partial w_{j,i}^{(1)}} = x_{n,i}.$$

Derivadas de la función de error (VIII)

- El valor de $\frac{\partial E_n(\mathbf{w})}{\partial a_{n,j}}$ depende de la forma de $E_n(\mathbf{w})$.
- Supongamos el problema de regresión múltiple. En este caso

$$E_n(\mathbf{w}) = \frac{1}{2} \sum_{k=1}^K (t_{n,k} - y_{n,k})^2,$$

donde

$$y_{n,k} = \sum_{j=0}^M w_{k,j}^{(2)} h(a_{n,j}),$$

- Luego,

$$\begin{aligned} \frac{\partial E_n(\mathbf{w})}{\partial a_{n,j}} &= \sum_{k=1}^K (t_{n,k} - y_{n,k}) w_{k,j}^{(2)} \frac{\partial h(a_{n,j})}{\partial a_{n,j}} \\ &= \frac{\partial h(a_{n,j})}{\partial a_{n,j}} \sum_{k=1}^K (t_{n,k} - y_{n,k}) w_{k,j}^{(2)}, \end{aligned}$$

Resumen derivadas de la función de error (RM)

Resumiendo

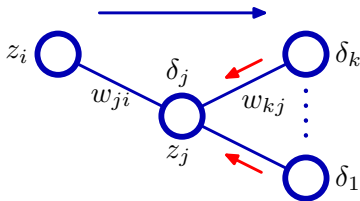
$$\frac{\partial E_n(\mathbf{w})}{\partial \mathbf{w}_{k,j}^{(2)}} = (y_{n,k} - t_{n,k}) z_{n,j} = \delta_{n,k} z_{n,j},$$

donde $\delta_{n,k} = t_{n,k} - y_{n,k}$.

Igualmente,

$$\frac{\partial E_n(\mathbf{w})}{\partial \mathbf{w}_{j,i}^{(1)}} = \frac{\partial h(a_{n,j})}{\partial a_{n,j}} \sum_{k=1}^K (t_{n,k} - y_{n,k}) \mathbf{w}_{k,j}^{(2)} x_{n,i} = \delta_{n,j} x_{n,i},$$

$$\text{donde } \delta_{n,j} = \frac{\partial h(a_{n,j})}{\partial a_{n,j}} \sum_{k=1}^K \delta_{n,k} \mathbf{w}_{k,j}^{(2)}.$$



Algoritmo general

1. Aplicar un vector de entrada \mathbf{x}_n a la red y propagarlo hacia adelante usando las ecuaciones para encontrar las activaciones de las unidades ocultas y las de salida.
2. Evaluar δ_k para todas las unidades de salida.
3. Propagar los δ 's del paso anterior para obtener los δ_j de cada nodo oculto en la red.
4. Usar $\frac{\partial E_n}{\partial w_{j,i}} = \delta_j z_i$ para evaluar las derivadas requeridas.

Ejemplo (I)

- Se considera una red de dos capas, el error viene dado como la suma de los errores cuadráticos.
- Las funciones de activación de salida son lineales $y_k = a_k$.
- Los nodos de ocultos tienen funciones de activación tangente hiperbólica

$$h(a) = \tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}.$$

Ejemplo (II)

- La derivada de la función se expresa como

$$\dot{h}(a) = 1 - h(a)^2.$$

- Para el patrón n , el error está dado como

$$E_n = \frac{1}{2} \sum_{k=1}^K (y_{n,k} - t_{n,k})^2.$$

- Para cada patrón de entrenamiento se realiza una propagación hacia adelante,

$$a_j = \sum_{i=0}^D w_{j,i}^{(1)} x_i,$$

$$z_j = \tanh(a_j),$$

$$y_k = \sum_{j=0}^M w_{k,j}^{(2)} z_j.$$

Ejemplo (II)

- Se calculan los δ_k para cada unidad de salida, usando

$$\delta_k = y_k - t_k.$$

- Luego se propaga hacia atrás estos δ 's para los nodos ocultos usando

$$\delta_j = (1 - z_j^2) \sum_{k=1}^K w_{k,j}^{(2)} \delta_k.$$

- Finalmente las derivadas con respecto a la primera capa y la segunda capa de pesos están dadas por

$$\frac{\partial E_n}{\partial w_{j,i}^{(1)}} = \delta_j x_i, \quad \frac{\partial E_n}{\partial w_{k,j}^{(2)}} = \delta_k z_j.$$

Contenido

Redes de propagación hacia adelante

Historia de las redes neuronales

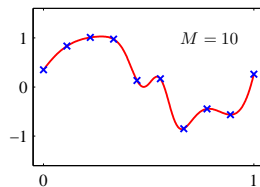
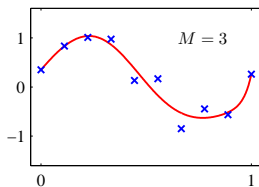
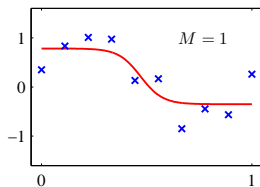
Entrenamiento de redes

Propagación del error hacia atrás

Regularización en redes neuronales

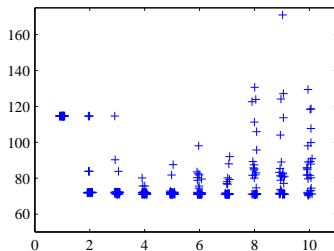
Generalidades (I)

- En una red neuronal el número de entradas está determinado por la dimensionalidad de los datos y el parámetro M es un parámetro libre que indica el número de nodos ocultos.
- El parámetro M determina el número de pesos y bias de la red.



Generalidades (II)

- ❑ Sin embargo, el error de generalización no es una función simple de M , debido a los mínimos locales de la función de error.
- ❑ En la figura se observa, que dependiendo de la inicialización, se obtienen diferentes valores del error cuadrático medio para un mismo valor de M .



Tipos de regularización (I)

- La regularización más simple tiene la forma

$$\tilde{E}(\mathbf{w}) = E(\mathbf{w}) + \frac{\lambda}{2} \mathbf{w}^\top \mathbf{w}$$

La complejidad del modelo se determina mediante λ .

- Se puede demostrar (ver Bishop, 2006) que este tipo de regularización no es invariante al re-escalamiento de los pesos y parámetros de sesgo de la red.

Tipos de regularización (II)

- Un tipo de regularización que sí es invariante a transformaciones lineales de las entradas o las salidas viene dada por

$$\frac{\lambda_1}{2} \sum_{w \in \mathcal{W}_1} w^2 + \frac{\lambda_2}{2} \sum_{w \in \mathcal{W}_2} w^2,$$

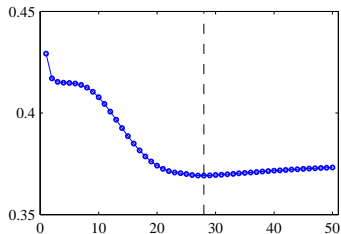
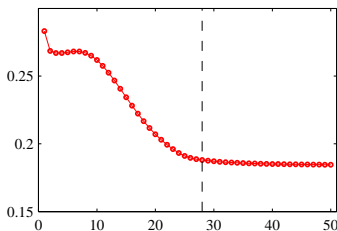
donde \mathcal{W}_1 denota el conjunto de pesos de la primera capa, y \mathcal{W}_2 denota el conjunto de pesos de la segunda capa.

Early stopping (Detención temprana) (I)

- Para muchos de los algoritmos de optimización usados para el entrenamiento de redes, el error es una función no creciente del índice de iteración.
- Sin embargo, para un conjunto de validación, el error medido decrece al principio y empieza a crecer cuando la red comienza a sobreentrenarse.

Early stopping (Detención temprana) (II)

- El entrenamiento puede entonces detenerse en el punto de menor error con respecto al conjunto de validación, con el fin de obtener una red con una buena capacidad de generalización.



Otras formas de regularización

Otras formas de regularización en redes neuronales incluyen (ver Bishop, 2006, secciones 5.5.4 a 5.5.6):

- ❑ Propagación de un vector tangente (tangent propagation).
- ❑ Entrenamiento con datos transformados.
- ❑ Redes convolucionales (convolutional networks).