uc3m | Universidad Carlos III de Madrid

OPENCOURSEWARE
REDES DE NEURONAS ARTIFICIALES
Inés M. Galván – José M. Valls

Tema 6

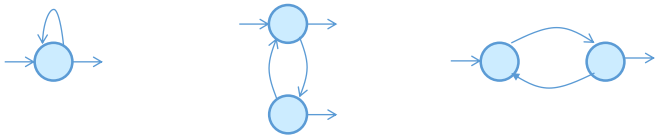
Redes de Neuronas Recurrentes

Redes Neuronales Recurrentes

- Introducción
- Red de Hopfield
- Redes parcialmente recurrentes
  - Red de Jordan
  - Red de Elman
- Redes totalmente recurrentes
- Long Short-Term Memory

Introducción

- Pueden tener ciclos o bucles en las conexiones (conexiones recurrentes)
- Conexiones recurrentes:
  - de una neurona con ella misma
  - entre neuronas de una misma capa
  - entre neuronas de una capa a una capa anterior



Introducción

- Al permitir conexiones recurrentes **aumenta** el número de pesos o de **parámetros ajustables** de la red
  - Aumenta la capacidad de representación
  - Se complica el aprendizaje
- Las activaciones no dependen sólo de las activaciones de la capa anterior sino también de la activación de cualquier otra neurona conectada a ella e incluso de su propia activación

Introducción

- Es necesario incluir la **variable tiempo** en la activación de la neurona. Si f es la función de activación, la activación de la neurona viene dada por:

$$a_i(t+1) = f\left(\sum_j w_{ji} \cdot a_j(t)\right)$$

Siendo a_j la activación de todas las neuronas que poseen conexión a la neurona i

- La variable tiempo hace que las redes tengan un **comportamiento dinámico o temporal**

- Dos formas de entender el modo de actuación y aprendizaje
  - Evolución de las activaciones de la red hasta alcanzar un punto estable
  - Evolución de las activaciones de la red en modo continuo

Introducción

- **Evolución hasta alcanzar un punto estable**

  Evolucionar la red (activaciones de sus neuronas), desde un estado inicial hasta conseguir que las activaciones de todas las neuronas de la red no se modifiquen: **punto estable**

  - Estado inicial: viene dado por el patrón de entrada
  - Estado estable: representa al patrón de salida de la red
  - La **Red de Hopfield** utiliza este mecanismo

Introducción

- **Evolución de las activaciones en modo continuo**
  - En cada instante disponemos de la salida de la red
  - La salida depende en cada instante t de las entradas en el instante anterior t-1
  - Dos tipos de aprendizaje
    - Aprendizaje por épocas
    - Aprendizaje en tiempo real o continuo: se aplica la ley de aprendizaje en cada instante (para cada patrón)
  - Redes parcialmente recurrentes: sólo tienen ciertas conexiones recurrentes
  - Redes totalmente recurrentes: no tienen restricciones
  - Todas usan **algoritmos supervisados** para el ajuste de parámetros

Introducción

- Su comportamiento dinámico facilita el tratamiento de **información temporal** o **patrones dinámicos:**
  - patrones que dependen del tiempo
  - el valor del patrón en un determinado instante depende de sus valores en instantes anteriores de tiempo
- Las redes recurrentes son apropiadas para tratar información temporal
- También pueden emplearse redes estáticas como MLP o RNBR
  - entrada de la red: secuencia temporal de valores en el pasado

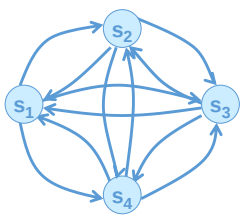
Introducción

Aplicaciones

- Problemas modelización neurobiólogica
- Tareas linguísticas
- Reconocimiento del palabras y fonemas
- Control de procesos dinámicos
- Predicción de series temporalis
- …

Red de Hopfield

- John Hopfield la propone en 1982 (Hopfield, 1982)
- Modelo de memoria asociativa
  - Es capaz de recuperar patrones almacenados a partir de información incompleta e incluso a partir de patrones con ruido
  - Actúa como memoria asociativa procesando patrones estáticos (sin variable tiempo)
  - Todas las neuronas están conectadas con todas las demás



Red de Hopfield

- Matriz de pesos W=(w_{ij}), orden n x n.
  - w_{ij} : peso de la conexión de neurona i a neurona j
  - Matriz simétrica $w_{ij} = w_{ji}$
  - Los elementos de la diagonal son nulos (no existen conexiones de una neurona a ella misma)
- Las neuronas poseen dos estados -1 y 1
- Estado de la neurona *i* en *t+1:*

$$s_i(t+1) = sgn(v_i(t+1)) \text{ para } i = 1, 2, ...n$$

$$sgn(v_i(t+1)) = \begin{cases} +1 & \text{si } v_i(t+1) > 0 \\ -1 & \text{si } v_i(t+1) < 0 \end{cases}$$

Red de Hopfield

- Donde $v_i(t+1)$ es el estado de activación de la neurona i calculado así:

$$v_i(t+1) = \sum_{j=1}^{n} w_{ji} s_j(t) - u_i \text{ para } i = 1, 2, ...n$$

donde $s_j(t)$ es el estado de la neurona j en el instante anterior t y u_i es un umbral fijo aplicado a la neurona i.

- Si el nivel $v_i(t+1) = 0$, se considera que el estado de de la neurona i no cambia
- No tiene sentido hablar de entradas o salidas sino del estado de la red en un intante t

Red de Hopfield

- Para una red con n neuronas, el estado en (t+1):

$$s(t+1) = [s_1(t+1), s_2(t+1), ..., s_n(t+1)]^t$$

- El estado s representa una palabra binaria con n bits de información

Aprendizaje y actuación de la red de Hopfiled

- Dos fases de operación
  - **Fase de almacenamiento**
    Se determinan los valores que tendrán los pesos para almacenar un conjunto de patrones
  - **Fase de recuperación**
    Mecanismo para recuperar la información almacenada a partir de información incompleta

Red de Hopfield. Fase de Almacenamiento

- Conjunto de patrones que se desea almacenar

$$\{x(k) = (x_1(k), x_2(k), ..., x_n(k))\}_{k=1,...,p}$$

donde cada patrón x(k) es un vector n-dimensional de componentes binarias (-1,1)

- Aplicando la regla de Hebb para almacenar patrones, el peso w_{ji} será:

$$w_{ji} = \sum_{k=1}^{p} x_j(k) x_i(k) \ \forall \ i \neq j$$

  - si las componentes $x_j(k)$, $x_i(k)$ son iguales, w_{ij} se incrementa en 1. Si son distintos, se decrementa en 1

Red de Hopfield. Fase de Recuperación

- Patrón de prueba $x = (x_1, x_2, ..., x_n)$ diferente de los patrones almacenados (versión incompleta o con ruido de algún patrón almacenado)
- La Red de Hopfield va a recuperar el patrón almacenado más parecido al patrón de prueba x
- Procedimiento:
  - Se inicializan los estados de las n neuronas de la red utilizando dicho patrón x:

$$s_i(0) = x_i \text{ para } i = 1, 2, ..., n$$

  - Se calculan los estados de la red en los siguiente instantes de tiempo utilizando las ecuaciones, hasta conseguir un **punto estable** (punto en el que los estados de las neuronas permanecen invariantes en el tiempo)
  - El estado estable de la red representa el patrón recuperado

Red de Hopfield. Fase de Recuperación

- Puede ocurrir que en la fase de recuperación la Red de Hopfiled converja a estados estables que no correspondan con los patrones almacenados
- Normalmente esto ocurre por almacenar un excesivo número de patrones

Red de Hopfield. Función Energía

- En MLP, RNBR, SOM, etc.. existe una **función de error o *energía*** que describe el funcionamiento de dichas redes
- En una Red de Hopfield con n neuronas la función que se minimiza al calcular los pesos es la llamada función de energía:

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j\neq i}^{n} w_{ij}s_i s_j + \sum_{i=1}^{n} u_i s_i$$

- Los puntos estables de la red se corresponden con mínimos de la función de energía

Capacidad de la red de Hopfield

- Cuestión planteada: ¿Cual es el máximo número de patrones que se pueden almacenar con seguridad en una red de Hopfield de dimensión determinada?
- Problema complicado y aún no resuelto de forma satisfactoria.
- La mayor parte de los estudios realizados ofrecen estimaciones asintóticas del máximo valor tolerable de patrones cuando la dimensión de la red es muy grande.
- Si se desea que todos los patrones sean recuperados con una probabilidad próxima a 1, se ha estimado la relación:

$$p \leq \frac{n}{4 \log n}$$

donde n es el número de neuronas y p el número de patrones que se desean recuperar.

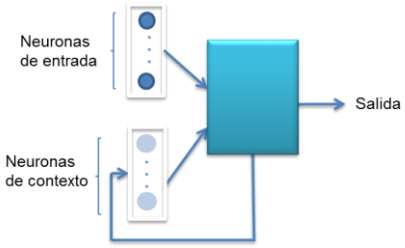
- Si se tolera un cierto margen de error en la recuperación de patrones, puede admitirse un número proporcional a la dimensión de la red, pudiendo llegar a ser del orden de 0.138 n.

Redes Parcialmente Recurrentes

- Son redes multicapa con algunas conexiones recurrentes
  - Estas conexiones permiten recordar el estado anterior de ciertas neuronas de la red
- Generalmente existen ciertas neuronas especiales en la capa de entrada: **neuronas de contexto**, receptoras de las conexiones recurrentes
  - Funcionan como una memoria de la red, almacenando el estado de las neuronas de una cierta capa en el instante anterior
- El resto de las neuronas de entrada actúan como receptores de los datos de entrada

Redes Parcialmente Recurrentes

- Concatenando las activaciones de las neuronas de entrada y de contexto, puede verse como un Percpetron Multicapa
- El cálculo de las activaciones se realiza como en una red multicapa sin recurrencias

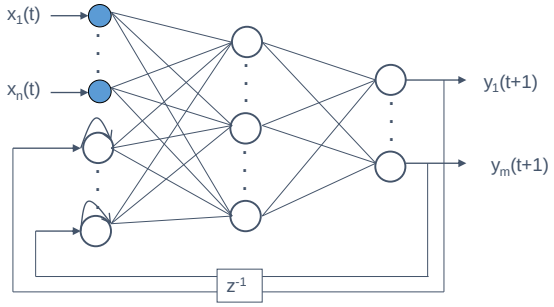


Redes Parcialmente Recurrentes

- Los pesos asociados a las conexiones recurrentes suelen ser constantes (no aprendizaje)
- Por tanto, se puede usar el algoritmo de retropropagación, ya que los únicos pesos ajustables son no recurrentes
- Las más conocidas
  - **Red de Jordan**
  - **Red de Elman**

Red de Jordan

- Propuesto por Jordan en 1986 (Jordan, 1986a,b)
- Las neuronas de contexto reciben una copia de las neuronas de salida y de ellas mismas
- Las conexiones recurrentes tienen un parámetro asociado μ (generalmente positivo y menor que 1)



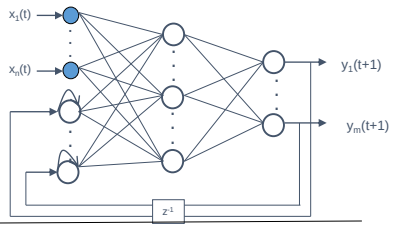
Red de Jordan

- Cada neurona de contexto recibe una conexión de una neurona de salida y de ella misma
- La activación de las neuronas de contexto en *t*:

$$c_i(t) = \mu c_i(t-1) + y_i(t-1) \text{ para } i = 1, 2, ..., m$$

- La entrada total de la red es una concatenación de las activaciones de las neuronas de entrada y de las neuronas de contexto

$$u(t) = (x_1(t), ..., x_n(t), c_1(t), ..., c_m(t))$$



Red de Jordan

- Neuronas ocultas y de salida: activación sigmoidal
- Neuronas de contexto: activación lineal
- Activación de las neuronas de contexto desarrollada en t:

$$c_i(t) = \mu c_i(t-1) + y_i(t-1) \text{ para } i = 1, 2, ..., m$$

$$c_i(t-1) = \mu c_i(t-2) + y_i(t-2)$$

$$c_i(t) = \mu^2 c_i(t-2) + \mu y_i(t-2) + y_i(t-1) = ... =$$

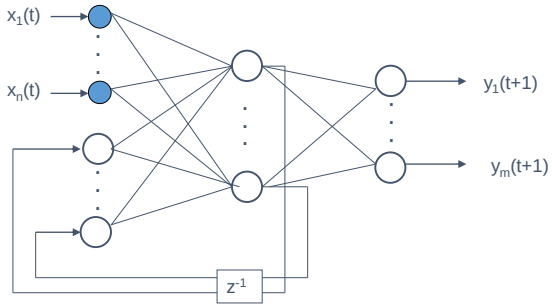
$$= \mu^{t-2} y_i(1) + \mu^{t-3} y_i(2) + ... + \mu y_i(t-2) + y_i(t-1)$$

Red de Jordan

- Las neuronas de contexto acumulan las salidas de la red para todos los instantes anteriores
- El valor de μ determina la sensibilidad de estas neuronas para retener dicha información
  - Si μ próximo a 0, entonces los estados alejados en el tiempo se olvidan con facilidad
  - Si μ próximo a 1, entonces los estados alejados en el tiempo se memorizan con facilidad

Red de Elman

- Propuesta por Elman en 1990 (Elman, 1990)
- Las neuronas de contexto reciben una copia de las neuronas ocultas de la red



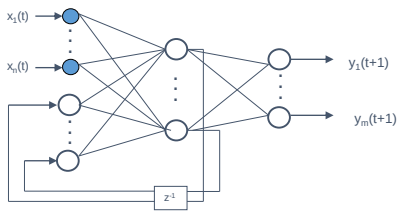
Red de Elman

- Tantas neuronas de contexto como ocultas
- No existe un parámetro asociado a la conexión recurrente
- La activación de las neuronas de contexto:

$$c_i(t) = a_i(t-1) \text{ para } i = 1, 2, ..., r$$

- El resto de las activaciones se calculan como en una red feedforward considerando como entrada el vector total:

$$u(t) = \left(x_1(t), ..., x_n(t), c_1(t), ..., c_m(t)\right)$$



Red de Elman y Jordan. Aprendizaje

1. Se inicializan las neuronas de contexto en t=0
2. En el instante t, se presenta el patrón $x(t) = (x_1(t), ..., x_n(t))$ que junto a las activaciones de las neuronas de contexto $c_i(t)$ forman el vector de entrada total de la red

$$u(t) = \big(x_1(t), ..., x_n(t), c_1(t), ..., c_m(t)\big)$$

3. Se propaga el vector u(t) hacia la salida de la red, obteniéndose la salida y(t)
4. Se aplica la regla delta generalizada para modificar los pesos de la red
5. Se incrementa la variable t en una unidad (t+1) y se vuelve al paso 2

Redes Totalmente Recurrentes

- No existe restricción de conectividad
- Sus neuronas reciben como entrada la activación del resto de neuronas y su propia activación

$$a_i(t) = f_i\Big(\sum_{j \in A \cup B} w_{ji} a_j(t-1)\Big)$$

A: neuronas de entrada

B: resto de neuronas

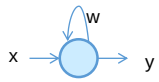
wji: peso de j a i

- Los pesos de las conexiones recurrentes se adaptan con aprendizaje
  - Aumenta el poder de representación
  - Se complica el aprendizaje

Redes Totalmente Recurrentes

- El algoritmo de retropropagación estándar no puede aplicarse a estas redes

Supongase la siguiente red:



$y(t+1) = f(x + wy(t))$ entonces $\frac{\partial y(t+1)}{\partial w} = w\frac{\partial y(t)}{\partial w}$

Teniendo en cuenta que $y(t) = f(x + wy(t-1))$, entontes $\frac{\partial y(t)}{\partial w} = w\frac{\partial y(t-1)}{\partial w}$ y así sucesivamente.

Por tanto las expresiones al calcular la derivada del error no son las mismaa que cuando se utiliza un PM.

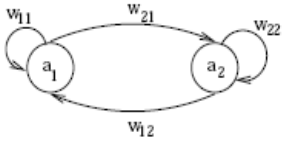
Redes Totalmente Recurrentes

- Existen dos algoritmos basados en retropropagación, adaptados para redes recurrentes:
  - Algoritmo de retropropagación a través del tiempo (Backpropagation through time) (Werbos, 1990)
  - Algoritmo de aprendizaje recurrente en tiempo real (Real time recurrent learning) (Williams and Zipser, 1989)

Algoritmo de retropropagación a través del tiempo

- Se basa en la idea de que para cada red recurrente es posible construir una red multicapa con conexiones hacia adelante. Basta desarrollar en el tiempo la red recurrente.

Sea por ejemplo la siguiente red:



Las activaciones vienen dadas por:

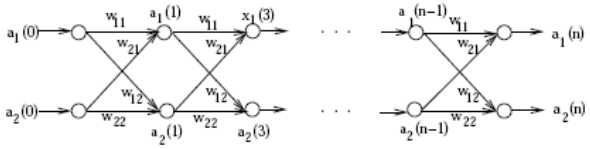
$$a_1(t) = f(w_{11}a_1(t-1) + w_{12}a_2(t-1))$$
$$a_2(t) = f(w_{21}a_1(t-1) + w_{22}a_2(t-1))$$

Algoritmo de retropropagación a través del tiempo

Desarrollando las activaciones de las dos neuronas en el tiempo, se obtiene:

$$a_i(1) = f(w_{i1}a_1(0) + w_{i2}a_2(0))\ a_i(2) = f(w_{i1}a_1(1) + w_{i2}a_2(1))$$
$$\ldots$$
$$a_i(n) = f(w_{i1}a_1(n-1) + w_{i2}a_2(n-1))$$

La red recurrente se puede representar mediante una red con conexiones hacia adelante (PM), añadiendo una nueva capa por cada unidad de tiempo:



Algoritmo de retropropagación a través del tiempo

- Dado el intervalo [n_0 n_1] en el se evoluciona la red, se define el error:

$$E(n_0, n_1) = \frac{1}{2} \sum_{n=n_0}^{n_1} \sum_{i \in S} e_i^2(n)$$

siendo S en conjunto de neuronas de salida

- El algoritmo de retro-propagacion a través del tiempo implica la aplicación del algoritmo de retro-propagacion a la red multicapa desarrollada en el tiempo.

- Es necesario tener en cuenta que:
  - Los pesos coinciden para todas las capas de la red multicapa desarrollada, por lo que al calcular el gradiente con respecto a un peso hay que hacerlo individualmente para cada capa y posteriormente sumarlos.

Algoritmo de retropropagación a través del tiempo

1. Dado un tiempo inicial n_0, la red recurrente se desarrolla en [n_0 n_1]

2. Se calculan las activaciones de todas las neuronas de la red multicapa.

3. Se calculan los valores $\boldsymbol{\delta}$ para cada una de las capas y neuronas de red: $\delta_i(n)$

4. Se ajustan los pesos de acuerdo con:

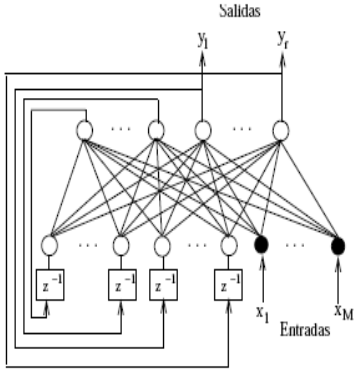
$$\Delta w_{ij} = \alpha \frac{\partial E(n_0, n_1)}{\partial w_{ij}} = \alpha \sum_{n=n_0+1}^{n_1} \delta_i(n) a_j(n-1)$$

5. Con los nuevos pesos, se repite el proceso para el instante de tiempo inicial n_0+ 1, preparando la red para una nueva época.

Algoritmo de aprendizaje recurrente en tiempo real

Sea la siguiente arquitectura de red recurrente:

- *M* neuronas de entrada y N neuronas en la red
- De las *N* neuronas, algunas serán las neuronas de salida (r)
- *M * N* conexiones hacia adelante
- *N * N* conexiones recurrentes
- Matriz de pesos: *W* = (*wij*) de orden *N x* (*M* + *N*).



Algoritmo de aprendizaje recurrente en tiempo real

- Sea *A* el conjunto de índices para las N neuronas de entrada y *B* para el resto de las neuronas
- Si *x*(*n*) = (x_i(n)) es el vector de entrada, se define el vector u(n) como:

$$u_i(n) = \begin{cases} x_i(n) & \text{si } i \in A \\ y_i(n) & \text{si } i \in B \end{cases}$$

- Las activaciones de las neuronas de salida de la red vienen dadas por:

$$y_i(n) = f(v_i(n-1)) \text{ para } i \in B$$
$$v_i(n-1) = \sum_{j \in A \cup B} w_{ij} u_j(n-1)$$

Algoritmo de aprendizaje recurrente en tiempo real

- El algoritmo de aprendizaje recurrente en tiempo real consiste en modificar los pesos de la red siguiendo la dirección negativa del gradiente del error.
- Para ello es necesario calcular las derivadas del error con respecto a cada peso.
- Si *S* el conjunto de índices para las neuronas de salida, el error se define como:

$$E(n) = \frac{1}{2}\sum_{i \in S} e_i^2(n)$$

donde $e_i(n)=(d_i(n)-y_i(n))$ $i \in S$

- La ley para modificar los pesos viene dada por:

$$w_{kl}(n) = w_{kl}(n-1) - \alpha\frac{\partial E(n)}{\partial w_{kl}} \text{ para } k \in B,\ l \in A \cup B$$

Algoritmo de aprendizaje recurrente en tiempo real

- Para calcular la derivada del error respecto a los pesos es necesario calcular la derivada de la salida respecto a los pesos
- Teniendo en cuenta la expresión de la activación de la salida, se obtiene:

$$\frac{\partial y_i(n)}{\partial w_{kl}} = f'\big(v_i(n-1)\big)\frac{\partial v_i(n-1)}{\partial w_{kl}}$$

- Por tanto:

$$\frac{\partial y_i(n)}{\partial w_{kl}} = f'(v_i(n-1)\sum_{j \in B} w_{ij}\frac{\partial y_j(n-1)}{\partial w_{kl}} + \delta_{ki}u_l(n-1)$$

siendo δ_{kl} la función delta de Kronecker, que vale 1 cuando *i* = *k* y 0 en cualquier otro caso.

Algoritmo de aprendizaje recurrente en tiempo real

- Por tanto:

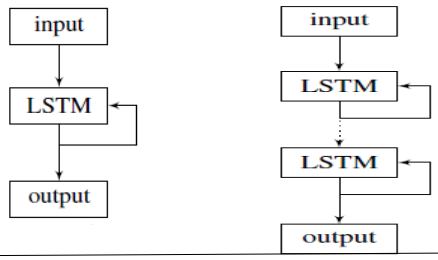
$$\frac{\partial E(n)}{\partial w_{kl}} = -\sum_{i \in S} e_i(n) p_{kl}^i(n)$$

$$p_{kl}^i(n) = f'(v_i(n-1)) \sum_{j \in B} w_{ij} p_{kl}^j(n-1) + \delta_{ki} u_l(n-1)$$

$$\text{para } k \in B, l \in A \cup B \text{ y } i \in S$$

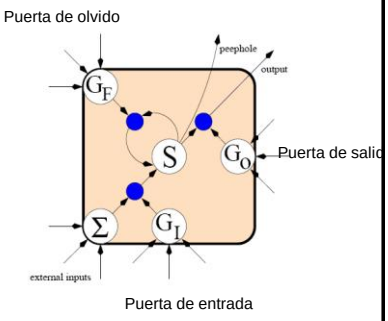
Long Sort-Term Memory (LSTM)

- Long short-term memory (LSTM) es una arquitectura de red recurrente propuesta por (Hochreiter and Schmidhuber, 1997)
- Está compuesta por unidades en la capa oculta, llamadas “memory blocks”.
- Cada bloque (LSTM) puede contener varias células
- Se pueden diseñar redes con varias capas LSTM



Long Sort-Term Memory (LSTM)

- Cada bloque contiene un determinado número de células o neuronas
- La activación de cada célula está controlada por tres puertas: entrada, olvido y salida
- Puerta de entrada: habilita/deshabilita la entrada
- Puerta de olvido: habilita/deshabilita el estado anterior
- Puerta de salida: habilita/deshabilita la salida



"LSTM memory block" con una célula

Long Sort-Term Memory (LSTM)

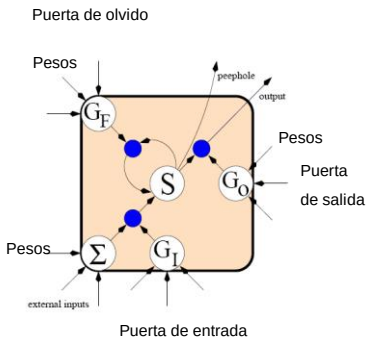
- Pesos:
  - De las entradas externas u otras células a la célula W^{cell}
  - De las entradas u otras células a las puertas W^{tipo}
- La salida de la célula viene dado por

$$c_i(t) = \tanh(g_i^{salida}(t) \cdot s_i(t))$$

donde tanh es la tangente hipérbólica;

g^{salida} la activación de la puerta de salida;

s_i el estado de la célula



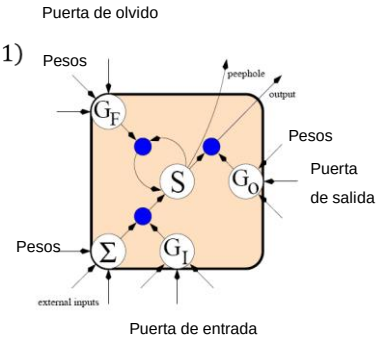
Long Sort-Term Memory (LSTM)

- El estado de la célula viene dado por:

$$s_i(t) = net_i(t) \cdot g_i^{entrada}(t) + g_i^{olvido}(t) \cdot s_i(t-1)$$

$g^{entrada}$ and g^{olvido} son las activaciones de las puertas de entrada y olvido

- Si la puerta de entrada está cerrada (activación $g_i^{entrada}$ cercana a cero), el estado de la célula no dependerá de la información que llegue a través de net_i.
- De manera similar, si la puerta de olvido está cerrada (activación g_i^{olvido} cercana a cero), el estado de la célula no dependerá del estado anterior.



net_i es la suma ponderada de las entradas externas (u_k) y activaciones de otras células (c_j)

$$net_i(t) = h\left(\sum_j w_{ij}^{cell} c_j(t-1)\right) + \sum_k w_{ik}^{cell} u_k(t)$$

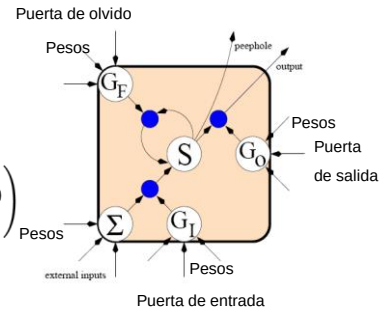
donde h suele ser la función identidad

Long Sort-Term Memory (LSTM)

- Las puertas reciben activaciones de otras células (c_j) o de entradas externas (u_k)

$$g_i^{tipo}(t) = \sigma\left(\sum_j w_{ij}^{tipo} c_j(t-1) + \sum_k w_{ik}^{tipo} u_k(t)\right)$$

Siendo σ la función de activación sigmoide



Long Sort-Term Memory (LSTM)

- Entrenamiento de LSTM: Minimizar el error en la salida de la red
  - Descenso del gradiente, como “backpropagation through time”. Los pesos cambian en proporción a la derivada del error con respecto al peso
  - Problema: “vanishing gradient problem”
  - Dificultad del aprendizaje
  - Alternativas:
    - Deep learning
    - Computación evolutiva

Referencias

(Hopfield, 1982) Hopfield, J. (1982). Neural Networks and physical systems with emergent collective computational abilities. In Proceedings of the National Academy of Science, vol 81, p 3088—3092. National Academy of Sciences

(Jordan, 1986a) Jordan, M. (1986). Attractor dynamics and parallelism in a connectionist sequential machine. In *Proc. of the Eighth Annual Conference of the Cognitive Science Society*, 531--546

(Jordan, 1986b) Jordan, M. (1986). Serial Order: a parallel distributed processing approach. Technical Report, Institute for Cognitive Science. University of California

(Elman, 1990) Elman J. (1990). Finding structure in time. *Cognitive Science*, 14: 179--211

(Werbos, 1990): Werbos, P. J. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, *78*(10), 1550-1560, 1990

(Williams and Zipser, 1989): Williams, R. J., & Zipser, D. (1989). Experimental analysis of the real-time recurrent learning algorithm. *Connection Science*, *1*(1), 87-111.

(Hochreiter and Schmidhuber, 1997): Hochreiter, S., & Schmidhuber, J. Long short-term memory. *Neural computation*, *9*(8), 1735-1780, 1997.