Capítulo 7

Inferencia estocástica con redes neurales

Trabajos recientes han demostrado que la extracción de reglas simbólicas mejora la potencia de generalización de las redes entrenadas con muestras completas de lenguajes regulares. En este capítulo, se explora la posibilidad de aprender reglas cuando la red se entrena con datos estocásticos. Para este fin, se utiliza una red con dos capas. Si en vez de utilizar la red misma, se extrae un autómata después del entrenamiento y sus probabilidades de transición se calculan a partir de la muestra, la distancia con respecto a la distribución correcta disminuye.

7.1 Antecedentes

Trabajos recientes como los de Giles *et al.* (1992a,b) y Watrous & Kuhn (1992a,b) han explorado la capacidad de las redes neurales de segundo orden para aprender gramáticas regulares sencillas a partir de muestras de entrenamiento completas. La potencia de generalización del autómata finito extraído a partir de la red entrenada es mejor que la de la misma red (Giles *et al.* 1992a,b; Omlin & Giles 1996). En la mayoría de los casos se observa que una vez que la red ha aprendido a

clasificar todas las palabras del conjunto de entrenamiento, los estados de las neuronas ocultas visitan durante el proceso de reconocimiento sólo algunas regiones o "clusters" del espacio de configuración accesible. Estos clusters pueden identificarse con los estados del autómata (DFA) inferido y las transiciones que entre ellos ocurren cuando se lee un símbolo en la entrada determinan la función de transición del DFA. Aunque la extracción de autómatas resulta aún controvertida (Kolen 1994), un trabajo reciente de Casey (1995) ha demostrado que una red recurrente de segundo orden puede modelizar robustamente un DFA si el espacio de configuración se divide en un varias zonas disjuntas asociadas a cada estado del autómata. Un algoritmo de extracción aparece descrito en Omlin & Giles (1996). La extracción se basa en dividir el hipercubo de configuración $[0.0, 1.0]^N$ en hipercubos más pequeños, de forma que cada uno contiene como mucho uno de los clusters. Un algoritmo mejorado que no restringe la forma de los clusters es el propuesto por Blair y Pollack (1996), pero su coste temporal es muy superior al anterior.

Debido a que las muestras completas son difíciles de obtener, es interesante estudiar si el mismo comportamiento puede ser observado cuando el entrenamiento se realiza a partir de muestras aleatorias. En ese caso, la extracción de una gramática subyacente debe mejorar la evaluación de las probabilidades, sobre todo para las cadenas no contenidas en la muestra.

Por ello, en este capítulo se explora la posibilidad de extraer reglas simbólicas a partir de la red entrenada con ejemplos generados aleatoriamente. La arquitectura de la red se describe en la sección siguiente y es análoga a una red de Elman (1990) con una función siguiente-estado de segundo orden, e idéntica a la usada por Blair y Pollack (1996). Una aproximación algorítmica al mismo problema es la presentada por Carrasco y Oncina (1994).

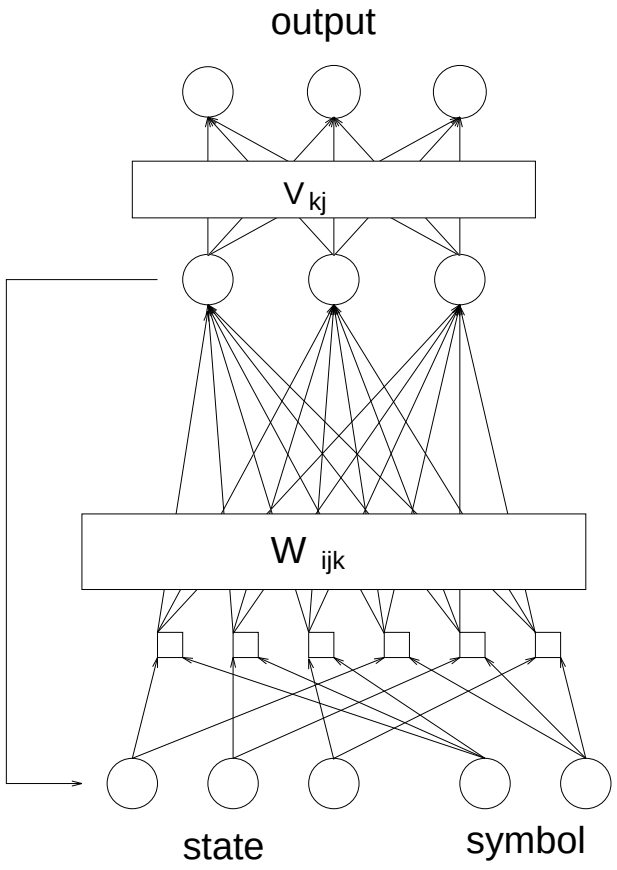

Figura 7.1: Red recurrente de segundo orden con $N = 3$ y $L = 2$

7.2 Arquitectura de la red

Supongamos que el alfabeto de entrada tiene L símbolos. Entonces, la red de segundo orden consta de N neuronas ocultas y L neuronas de entrada más $L + 1$ neuronas de salida tal y como aparece en la figura 7.1). Sus estados respectivos se etiquetan como S_k, I_k y O_k.

La red sólo lee un carácter por ciclo (tomamos $t = 0$ para el primer ciclo) y el vector de entrada es $I_k = \delta_{kl}$ cuando se procesa a_l, el l-ésimo

símbolo del alfabeto[1]. La salida $O_k^{[t+1]}$ representa la probabilidad de que a la cadena le siga el símbolo a_k para $k = 1, ...L$ y $O_{L+1}^{[t+1]}$ representa la probabilidad de que la cadena termine en ese punto. La dinámica de la red está regida por las ecuaciones

$$O_k^{[t]} = g(\sum_{j=1}^{N} V_{kj}\ S_j^{[t]}) \tag{7.1}$$

$$S_i^{[t]} = g(\sum_{j=1}^{N}\sum_{k=1}^{L} W_{ijk}\ S_j^{[t-1]}\ I_k^{[t-1]}) \tag{7.2}$$

donde $g(x) = 1/(1 + \exp(-x))$.

El entrenamiento de la red se realiza con una versión del método de Williams y Zipser (1989) conocido como "Real Time Recurrent Learning" (RTRL). La contribución de cada cadena w de la muestra a la energía (o función de error) viene dada por

$$E_w = \sum_{t=0}^{|w|} E_w^{[t]} \tag{7.3}$$

donde

$$E_w^{[t]} = \frac{1}{2}\sum_{k=1}^{L+1}\left(O_k^{[t]} - I_k^{[t]}\right)^2 \tag{7.4}$$

y $I_k^{[|w|]} = \delta_{k,L+1}$. Es sencillo probar que esta energía presenta un mínimo cuando $O_k^{[t]}$ es la probabilidad condicionada de obtener el símbolo a_k después de leer los primeros t símbolos de w. Dado que una probabilidad igual a cero (o uno) es difícil de aprender para la red, se usará una función contractiva de la entrada $f(x) = \epsilon + (1 - 2\epsilon)x$, con ϵ un pequeño número real positivo (o equivalentemente, una expansión de la salida). Esta contracción no afecta esencialmente al formalismo que aquí se presenta.

Normalmente, los pesos W_{ijk} son inicializados con valores pequeños y después son optimizados, pero los valores iniciales $S_i^{[0]}$ de las neuronas ocultas no cambian durante el entrenamiento. Tal y como se

[1]La delta de Kronecker δ_{kl} es 1 si $k = l$ y 0 en caso contrario.

detalla en Forcada & Carrasco (1995), no hay ninguna razón para forzar a estos valores iniciales a permanecer fijos, y el comportamiento de la red mejora si también se permite su aprendizaje. Este detalle resulta aquí de gran importancia, pues la probabilidad predicha por la red $O_k^{[t]}$ depende de los estados $S_i^{[t]}$, mientras que en una tarea de clasificación es suficiente con que los estados de aceptación se encuentren en una zona (la zona de aceptación) y los demás en otra zona (la de no aceptación). Con el fin de que los valores de $S_i^{[t]}$ permanezcan dentro del rango permitido $[0.0, 1.0]$ durante el descenso por gradiente, es mejor definir las entradas de la red $\sigma_i^{[t]}$ tales que $S_i^{[t]} = g(\sigma_i^{[t]})$ y tomar $\sigma_i^{[0]}$ como los parámetros que hay que aprender, ya que estos pueden tomar cualquier valor.

Por tanto, todos los parámetros V_{kj}, W_{ijk} y $\sigma_i^{[0]}$ son inicializados con pequeños valores aleatorios antes del entrenamiento, pero después se aprenden simultáneamente. Cada parámetro P se actualiza de acuerdo con el descenso por gradiente

$$\Delta P = -\alpha \frac{\partial E}{\partial P} + \eta \Delta' P \tag{7.5}$$

con una tasa de aprendizaje α y un término de momento η. El valor $\Delta' P$ representa la variación del parámetro P en la iteración anterior.

La contribución de cada símbolo de w a la derivada de la energía es:

$$\frac{\partial E_w^{[t]}}{\partial P} = \sum_{b=1}^{L+1} (O_b^{[t]} - I_b^{[t]}) O_b^{[t]} (1 - O_b^{[t]}) \sum_{a=1}^{N} (\frac{\partial V_{ba}}{\partial P} S_a^{[t]} + V_{ba} S_a^{[t]} (1 - S_a^{[t]}) \frac{\partial \sigma_a^{[t]}}{\partial P}) \tag{7.6}$$

donde se ha utilizado la siguiente propiedad de la derivada de la función sigmoidea: $g'(x) = g(x)(1 - g(x))$.

Las derivadas con respecto a V_{kj} pueden ser calculadas inmediatamente:

$$\frac{\partial E_w^{[t]}}{\partial V_{kj}} = (O_k^{[t]} - I_k^{[t]}) O_k^{[t]} (1 - O_k^{[t]}) S_j^{[t]} \tag{7.7}$$

Sin embargo, las derivadas con respecto a $\sigma_i^{[0]}$ y W_{ijk} deben ser calcu-

ladas de forma recurrente, usando las ecuaciones siguientes:

$$\frac{\partial \sigma_m^{[0]}}{\partial W_{ijk}} = 0 \tag{7.8}$$

$$\frac{\partial \sigma_i^{[0]}}{\partial \sigma_j^{[0]}} = \delta_{ij} \tag{7.9}$$

$$\frac{\partial \sigma_i^{[t]}}{\partial \sigma_j^{[0]}} = \sum_{a=1}^{N} S_a^{[t-1]}(1 - S_a^{[t-1]}) \sum_{b=1}^{L} W_{iab} \frac{\partial \sigma_a^{[t-1]}}{\partial \sigma_j^{[0]}} I_b^{[t-1]} \tag{7.10}$$

$$\frac{\partial \sigma_m^{[t]}}{\partial W_{ijk}} = \delta_{im} S_j^{[t-1]} I_k^{[t-1]} + \sum_{a=1}^{N} S_a^{[t-1]}(1 - S_a^{[t-1]}) \sum_{b=1}^{L} W_{mab} \frac{\partial \sigma_a^{[t-1]}}{\partial W_{ijk}} I_b^{[t-1]} \tag{7.11}$$

7.3 Resultados y discusión

El comportamiento de la red fue estudiado usando la siguiente gramática regular estocástica:

$$\begin{array}{lcll} S & \rightarrow & 0S & (0.2) \\ S & \rightarrow & 1A & (0.8) \\ A & \rightarrow & 0B & (0.7) \\ A & \rightarrow & 1S & (0.3) \\ B & \rightarrow & 0A & (0.4) \\ B & \rightarrow & 1B & (0.1) \end{array} \tag{7.12}$$

donde los números entre paréntesis indican la probabilidad de las reglas. La probabilidad de final de cadena para una variable puede calcularse como la diferencia entre 1 y la suma de las probabilidades de todas las reglas que contienen a dicha variable en la parte izquierda. Se usaron dos neuronas de entrada ($L = 2$) y tres de salida (asociadas a 0, 1 y final de cadena respectivamente), mientras que se tomó $N = 4$ neuronas ocultas. La red fue entrenada con 800 ejemplos aleatorios[2] y

[2]La muestra contenía 200 cadenas diferentes en promedio.

todas ellas fueron usadas en todas las iteraciones. La tasa de aprendizaje se eligió $\alpha = 0.004$ y el término de momento $\eta = 0.2$. El proceso de entrenamiento para cada muestra duró 1000 iteraciones. La figura 7.2

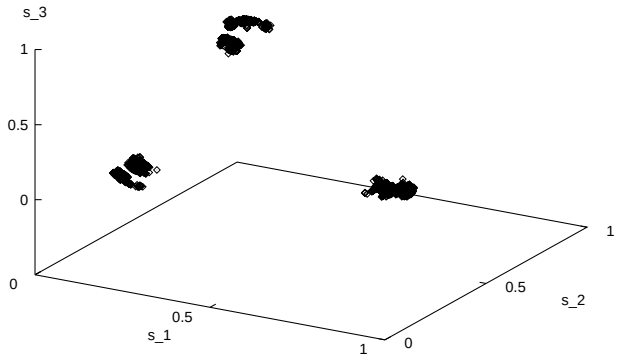

Figura 7.2: Espacio de configuración después del entrenamiento con 800 ejemplos.

muestra un a sección típica del espacio de configuración $[0.0, 1.0]^N$ de una red entrenada. Cada punto representa un vector $S^{[t]}$, generado para todas las palabras de longitud menor o igual que 10. Tal y como se ve en la figura, sólo algunas regiones del espacio son visitadas y se observan tres clusters asociados a las tres variables de la gramática. La aparición de clusters se produjo en todos los experimentos.

En la figura 7.3, se representa la proyección al espacio de salida (O_1, O_2, O_3). Los clusters se proyectan sobre tres regiones centradas en los puntos (0.2,0.8,0.0), (0.7,0.3,0.0) y (0.4,0.1,0.5), de acuerdo con las probabilidades correctas. Debe notarse que, a pesar de no ser una restricción impuesta a la red, las probabilidades suman uno aproximadamente. Debido a la contracción utilizada (se tomó $\epsilon = 0.1$), algunos valores aparecen ligeramente por debajo de cero y deben ser tratados

como imprecisiones o errores de redondeo. La figura demuestra que la red es capaz de predecir la probabilidad del siguiente símbolo con gran precisión.

Una medida de la similitud entre dos distribuciones de probabilidad, en nuestro caso la distribución $q(w)$ calculada por el modelo y las probabilidades $p(w)$ correctas es la llamada distancia de Kullback-Leibler o entropía relativa (Cover & Thomas 1991):

$$d(p,q) = \sum_w p(w) \log_2 \frac{p(w)}{q(w)} \tag{7.13}$$

cuyo valor promedio en los experimentos realizados fue de 0.02 bits. En la tabla siguiente se muestran los resultados en diez experimentos para la distancia $d(A,S)$ entre el autómata correcto A y el conjunto de muestra S, la distancia entre A y la distribución evaluada mediante la red recurrente entrenada N y, finalmente, la distancia entre el autómata correcto A y el extraído de la red A'.

exp.	$d(A,S)$	$d(A,N)$	$d(A,A')$
1	1.255	0.0337587	0.00612741
2	1.196	0.0124381	0.00203993
3	1.198	0.0374824	0.00370043
4	1.309	0.0445123	0.00140762
5	1.276	0.0230212	0.00153648
6	1.196	0.0016789	0.00142349
7	1.306	0.0187031	0.0138799
8	1.405	0.0063514	0.00716541
9	1.288	0.0134842	0.00203797
10	1.495	0.0126591	0.00415552

Se puede observar, que si el autómata era extraído de la red y se calculaban las probabilidades de transición a partir de la muestra, la distribución $q(w)$ generada por este autómata se encontraba a una distancia $d(p,q)$ promedio de 0.004 bits, mucho menor que el valor 0.02 obtenido a partir de la red directamente. Dado que en todos los experimentos aparecieron los tres cúmulos asociados a los tres estados

del autómata, este resultado es igual al que se obtiene con el algoritmo `rlips`, que también en todos los casos identificó la estructura correcta del autómata.

Si tenemos en cuenta, que la distancia entre la muestra y el autómata correcto era en promedio 1.2 bits, es posible afirmar que el autómata extraído a partir de la red mejora significativamente el rendimiento de la red en la predicción de las probabilidades de las cadenas no observadas, aunque sus resultados no mejoren el rendimiento de `rlips`. De hecho, experimentos realizados con otras gramáticas demostraron que no siempre es tan sencilla la formación de cúmulos en el espacio de configuración. Por ejemplo, la gramática

$$\begin{array}{lcll} S & \rightarrow & 0S & (0.5) \\ S & \rightarrow & 1A & (0.5) \\ A & \rightarrow & 0A & (0.1) \\ A & \rightarrow & 1B & (0.4) \\ B & \rightarrow & 0B & (0.3) \\ B & \rightarrow & 1S & (0.3) \end{array} \tag{7.14}$$

requiere la utilización de una red con cinco neuronas. Con muestras de 1000 ejemplos, se produjo la identificación en tres de diez experimentos, con una entropía relativa en estos casos de 0.38 bits (análoga a la de `rlips`). En el resto de los casos no aparecieron cúmulos, lo que produjo una entropía relativa infinita, en contraste con el buen funcionamiento de `rlips` en todos los casos. Merece la pena destacar que a veces se observa la aparición de atractores no puntuales, como el representado en la figura 7.4. Este atractor toroidal demuestra por un lado la existencia de una dinámica compleja en la red cuyo estudio puede resultar de interés. Por otro lado, es un posible indicación de la insuficiencia de los métodos de retro-propagación para obtener con la exactitud requerida el mínimo de la función de error. Una línea de investigación que abordaremos próximamente es el desarrollo o aplicación de otros métodos de entrenamiento para este tipo de redes, como por ejemplo el `ALOPEX` de Unnikrishnan y Venugopal (1994).

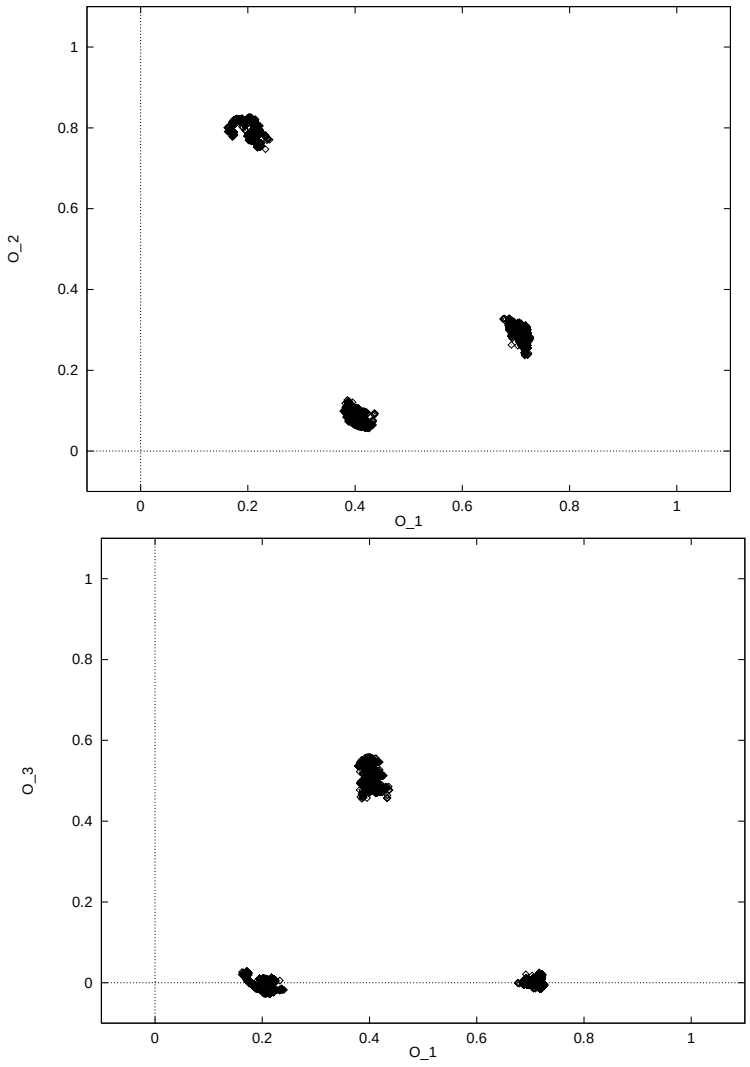

Figura 7.3: Probabilidades de salida después del entrenamiento

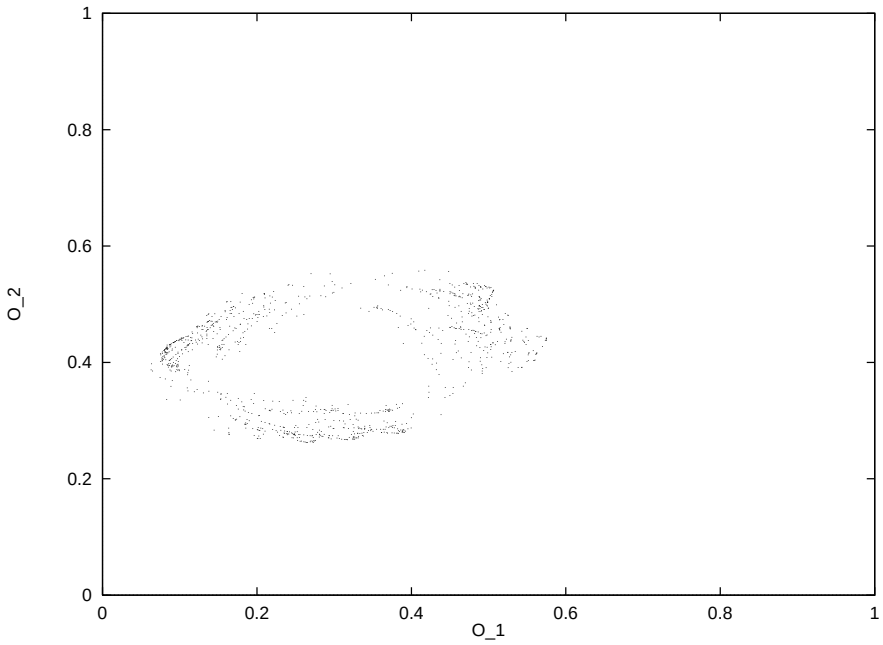

Figura 7.4: Atractor de tipo toroidal obtenido durante el entrenamiento de una red con $N = 5$ para la gramática 7.14