Esquema de Aprendizaje Basado en la Ecuación HJB para Redes Neuronales

Esteban Reyes Saldaña[1], Dra. Ligia Quintana Torres[2], Dr. Porfirio Toledo Hernández[3]

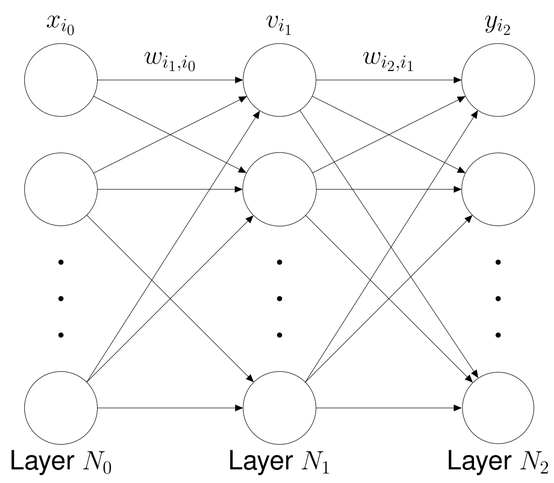
Universidad Veracruzana, Facultad en Matemáticas

estebanrs_28@hotmail.com[1], liquintana@uv.mx[2], portoledo@gmail.com [3]

1. Definiciones

Considérese una red neuronal FFNN (Feed Forward Neural Network), con estructura $N_0 - N_1 - N_2$. Se introduce la no linealidad mediante la siguiente función de activación sigmoide

$$h_{i_1} = \sum_{i_0} w_{i_1 i_0} x_{i_0} \ , \ v_{i_1} = \frac{1}{1+e^{-h_{i_1}}},$$
$$h_{i_2} = \sum_{i_1} w_{i_2 i_1} v_{i_1} \ , \ y_{i_2} = \frac{1}{1+e^{-h_{i_2}}}.$$



Con esta estructura, la red neuronal define una función continua no lineal del $\mathbb{R}^{N_0}$ (espacio de los patrones de entrada) al espacio $\mathbb{R}^{N_2}$ (espacio de los patrones de salida). Por lo tanto, para cada $X_p \in \mathbb{R}^{N_0}$ y cada $\widehat{w} \in \mathbb{R}^{N_w}$ se obtiene un vector de salida $Y_p \in \mathbb{R}^{N_2}$.
De esta manera podemos escribir

$$Y_p = f(\widehat{w}, X_p),$$

dónde $\widehat{w}$ es un vector de todos los pesos involucrados en la red, con un total de N_w parámetros. Además, para la salida deseada Y_p^d, existe w tal que $Y_p^d = f(w, X_p)$.
El objetivo de la red es minimizar el error que está dado por

$$e_p = Y_p^d - Y_p.$$

Derivando con respecto al tiempo:

$$\dot{e}_p = \dot{Y}_p^d - \dot{Y}_p = 0 - \frac{\delta f(\widehat{w}, X_p)}{\delta \widehat{w}} \dot{\widehat{w}} = -J_p \dot{\widehat{w}} \qquad (1)$$

dónde J_p es la matriz jacobiana y $\dot{Y}_p^d = 0$.

2. Actualización de Pesos Óptima

La optimización de los pesos de la red neuronal se formula como un problema de control de la siguiente manera

$$\dot{e}_p = -J_p \dot{\widehat{w}} = -J_p u$$

en donde $e = (e_1^T, e_2^T, \ldots, e_{N_p}^T)^T$ es un vector $N_l N_p \times 1$ y $J = (J_1^T, J_2^T, \ldots, J_{N_p}^T)^T$ es una matriz $N_l N_p \times N_w$. Dado un vector u de tamaño $N_w \times 1$, la función de costo en $(t, T]$ es

$$V(e(t)) = \int_t^T L(e(s), u(s)) ds, \qquad (2)$$

en donde

$$L(e, u) = \frac{1}{2} e^T e + u^T R u, \qquad (3)$$

con R matriz identidad de tamaño $N_w \times N_w$ y t el número de iteraciones en la actualización de pesos.

META: encontrar $u(t)$ que optimice $V(e(t))$. Se asume que $T \to \infty$.

Ecuación HJB

$$\min_u \left\{ t \frac{dV^*}{de} \dot{e}(t) + L(e(t), u(t) \right\} = 0.$$

De las expresiones (1), (2) y (3) se tiene que

$$\min_u \left\{ -\frac{dV^*}{de} J u(t) + \frac{1}{2} e(t)^T e(t) + \frac{1}{2} u(t)^T R u(t) \right\} = 0.$$

Diferenciando con respecto a u, obtenemos la ley de pesos óptima

$$u^*(t) = R^{-1} J^T \left(\frac{dV^*}{de} \right)^T.$$

Así que

$$e(t)^T e(t) - \left(\frac{dV^*}{de} \right) J R^{-1} J^T \left(\frac{dV^*}{de} \right)^T = 0. \qquad (4)$$

Una solución propia a esta ecuación debe dejar a $JR^{-1}J^T$ definida positiva.

Función de Liapunov

El resultado óptimo se alcanzará cuando la solución sea estable. Se usa la función de Liapunov:

$$\nu(e) = \frac{1}{2} e^T e \qquad (5)$$

donde $e = 0$ es un punto de equilibrio estable si $\dot{\nu}(e)$ se define negativa.

$$\begin{aligned} \dot{\nu}((e(t)) &= e^T e \\ &= -e^T J u^*(t) \\ &= -e^T J R^{-1} J^T \left(\frac{dV^*}{de} \right)^T. \end{aligned}$$

Si se escoge dV^*/de como

$$\frac{dV^*}{de} = e(t)^T C(t)^T, \qquad (6)$$

donde $C(t)$ está definida positiva. Así que $\dot{\nu}(e(t))$ queda definida negativa. Para encontrar la expresión de $C(t)$ sustituímos la ecuación (6) en la expresión (4). Luego

$$e(t)^T (I - C(t)^T J R^{-1} J^T C(t)) e(t) = 0. \qquad (7)$$

Donde I es la matriz identidad de tamaño $N_p \times N_p$. Para que esto sea verdad para todo $e(t)$ se debe cumplir

$$C(t)^T J R^{-1} J^T C(t) = I. \qquad (8)$$

Para encontrar una solución para $C(t)$ descomponemos $JR^{-1}J^T = U \Sigma U^T$ donde U es una matriz de eigenvalores y Σ es una matriz diagonal de eigenvalores. Así que $C(t)$ satisface

$$C(t) = U \Sigma^{-\frac{1}{2}} U^T. \qquad (9)$$

Finalmente, obtenemos la ley de pesos óptima como

$$\dot{\widehat{w}} = u^*(t) = R^{-1} J^T C(t) e(t). \qquad (10)$$

Para la estructura de la FFNN el jacobiano se puede obtener como sigue
Para la capa de salida:

$$\frac{\partial f(w, x_p)}{\partial w_{i_2, i_1}} = y_{p,i_2}(1 - y_{p,i_2}) v_{i_1}.$$

Para la capa oculta:

$$\frac{\partial f(w, x_p)}{\partial w_{i_1, i_0}} = \sum_{i_2} y_{p,i_2}(1 - y_{p,i_2}) w_{i_2,i_1} v_{i_1}(1 - v_{i_1}) x_{p,i_0}.$$

3. Resultados

Se tomó la función módulo-2, $f : \mathbb{R}^2 \to \mathbb{R}$. Donde las posibles combinaciones se obtienen usando $x_1 \in \{0, 1, 2\}$ y $x_2 \in \{0, 1, 2\}$. Los resultados esperados están dados por

$$f(x_1, x_2) = \begin{cases} 1, & \text{si } x_1 + x_2 \text{ es impar,} \\ 0, & \text{si } x_1 + x_2 \text{ es par.} \end{cases}$$

Los algoritmos se implementaron en MATLAB R2013a en Intel(R) Xeon (R), 2.40 GHz CPU con 6GB RAM. Cada experimento consiste en múltiples intentos, cada uno iniciando desde un punto aleatorio. Para una comparación justa, se eligieron los mismos puntos iniciales para cada esquema de aprendizaje. La simulación se realizó para 25 intentos diferentes.

(a) algoritmos offline

FFNN arquitectura	BP $\eta = 1$	LF $\mu = 0.05$	HJB $r = 0.1$
3-4-1	3047 (7.6 s)	1087 (9.5 s)	3519 (6.0 s)
3-6-1	2880 (4.2 s)	2801 (9.9 s)	190 (0.4 s)
3-8-1	1309 (2.4 s)	480 (10.4 s)	134 (0.5 s)
3-10-1	1046 (1.4 s)	447 (0.7 s)	54 (0.09 s)
3-15-1	2544 (3.5 s)	670 (0.9 s)	49 (0.07 s)

(b) algoritmos online

FFNN arquitectura	LM $\eta = 10^{-2}$, $\beta = 10$	HJB-LM $\eta = 10^{-2}$, $\beta = 10$
3-4-1	266 (7.0 s)	63 (0.3 s)
3-6-1	37 (0.15 s)	25 (0.05 s)
3-8-1	22 (0.06 s)	26 (0.06 s)
3-10-1	18 (0.05 s)	22 (0.05 s)
3-15-1	17 (0.05 s)	25 (0.05 s)

4. Conclusiones

En este cartel se plantea el problema de la actualización instantánea de los pesos de una red neuronal FFNN como un problema de control. Se utiliza la ecuación de Hamilton-Jacobi-Bellman (HJB) para encontrar una regla de actualización de pesos óptima. La contribución principal de este cartel es que, utilizando la ecuación HJB, pueden obtenerse soluciones tanto para el costo óptimo como para las actualizaciones de los pesos en cualquier red neuronal Feed-Forward. Se compara el enfoque propuesto con algunos de los mejores algoritmos de aprendizaje que existen. Se ha encontrado que con esta propuesta la convergencia es más rápida en términos de tiempo computacional.

Referencias

[1] ARORA, V., BEHERA, L., REDDY, T. K., AND YADAV, A. P. HJB equation based learning scheme for neural networks. In *2017 International Joint Conference on Neural Networks (IJCNN)* (May 2017), pp. 2298–2305.

[2] BEALE, R., AND JACKSON., T. *Neural Computing - An Introduction*. Taylor & Francis, 1990.