UNIVERSIDAD DE BUENOS AIRES
FACULTAD DE INGENIERÍA

MAESTRÍA EN SIMULACIÓN NUMÉRICA Y CONTROL

Trabajo práctico 3:

MONOGRAFÍA

Redes Neuronales Artificiales para relaciones constitutivas aplicadas al Método de Elementos Finitos

Curso:

Sistemas adaptativos y Redes Neuronales

Estudiante:
Fredy Andrés Mercado Navarro
DNI: 94.872.342

Profesor titular:
Sergio Lew

Clases Prácticas:
Ana Laura Vadnjal

1 de mayo de 2014
Buenos Aires, Argentina

Resumen

Esta monografía es la aplicación de una red neuronal, el Perceptrón, aplicada a la estimación numérica de relaciones constitutivas para ser aplicadas en calculos estructurales. Se empleará el Perceptrón Multicapa con Backpropagation para procesar datos de entrada y salida, y lograr pesos sinápticos que logren relacionar efectivamente deformaciones y tensiónes para modelos de material lineales y no lineales. Este trabajo mostrará la capacidad de este tipo de Redes Neuronales Artificiales - RNAs, para reemplazar las relaciones constitutivas matriciales clásicas y lograr una mejora en los modelos que se encargan de predecir la respuesta de materiales con comportamientos no lineales complejos de modelar, como son los compuestos que utilizan fibra de vidrio y fibra de carbono. Las redes serán entrenadas a partir de datos sintéticos que harán las veces de datos experimentales, buscando llegar a una red que permita estimar acertadamente un estado de tensiones a partir de un estado de deformaciones.

Índice general

Índice de figuras

Capítulo 1

Planteamiento del Problema

Para modelar el comportamiento mecánico de materiales se emplean relaciones constitutivas para asociar el cambio de una variable con otra. El caso más representativo es el del tensor constitutivo de orden 4 que relaciona el tensor de deformaciones con el tensor de tensiones. Esta relación está definida en un punto de un material. Si empleamos la notación de Voight, estos tensores se pueden relacionar como

$$\sigma = C\varepsilon \tag{1.1}$$

donde σ es un vector que contiene las componentes que definen el estado de tensiones, C es la matriz constitutiva que relaciona deformaciones y tensiones y ϵ es el vector que contiene las componentes que definen el estado de deformaciones en el punto considerado. Para el caso tridimensional, es decir, el caso más general, tenemos 6 componentes para las tensiones y las deformaciones, y 36 para la relación constitutiva C. La ecuación 1.1 se puede escribir entonces como

$$\begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \tau_{12} \\ \tau_{23} \\ \tau_{13} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\ C_{21} & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} \\ C_{31} & C_{32} & C_{33} & C_{34} & C_{35} & C_{36} \\ C_{41} & C_{42} & C_{43} & C_{44} & C_{45} & C_{46} \\ C_{51} & C_{52} & C_{53} & C_{54} & C_{55} & C_{56} \\ C_{61} & C_{62} & C_{63} & C_{64} & C_{65} & C_{66} \end{bmatrix} \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ \gamma_{12} \\ \gamma_{23} \\ \gamma_{13} \end{bmatrix} \tag{1.2}$$

donde σ y τ son tensiones normales y de corte, y ε y γ son deformaciones normales y de corte, respectivamente.

Las relaciones constitutivas tienen el propósito de aproximar lo mejor posible el comportamiento natural de un material sometido a cargas o desplazamientos. Estas se han desarrollado con base en comparaciones entre los modelos matemáticos y los resultados experimentales, los cuales le han dado validez a los modelos más sencillos, sin embargo, cuando se desea modelar la respuesta de materiales no lineales se requieren de relaciones constitutivas que capturen las características no lineales del material, lo cual puede ser provocado por la dependencia de más variables aparte de la deformación, como por ejemplo, el endurecimiento que sufren los metales con la deformación.

Se propone entonces realizar una aplicación de la teoría del Perceptrón con Backpropagation para entrenar una red a partir de datos sintéticos que consideraremos ex-

perimentales, y una vez entrenada procederemos a comprobar la calidad de la relación constitutiva encontrada y a tratar de validarla para un conjunto de datos de entrada.

Capítulo 2

Perceptrón Multicapa con Backpropagation

El perceptrón es una Red Neuronal de aprendizaje supervisado que transforma una entrada en una salida, pudiendo aprender funciones de gran complejidad a través del algoritmo de propagación del error hacia atrás (backpropagation). A continuación se presenta un resumen ilustrativo, teniendo en cuenta que son contenidos que se han estudiado anteriormente en el curso de Redes Neuronales. La notación que se emplea en este resumen puede ser consultada en la referencia [8].

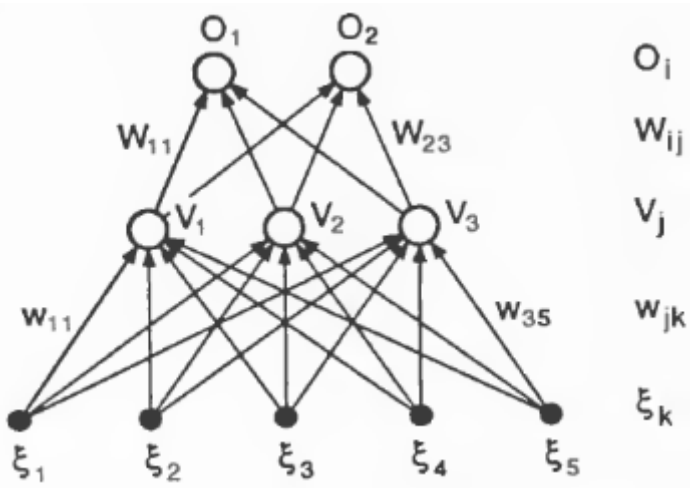

Figura 2.1: Red Neuronal Feed-Forward de 2 capas. Tomada de [8].

Dado un patrón μ, la unidad escondida j recibe una entrada neta

$$h_j^\mu = \sum_k w_{jk} \xi_k^\mu$$

Si Suponemos que tenemos una red de 4 entradas, una capa oculta con 20 neuronas y 4 salidas podemos expresar esto como:

$$\begin{bmatrix} h_1^\mu \\ h_2^\mu \\ . \\ . \\ . \\ h_{20}^\mu \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & . & w_{14} \\ w_{21} & w_{22} & . & w_{24} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ w_{20,1} & . & . & w_{20,4} \end{bmatrix} \begin{bmatrix} \xi_1^\mu \\ \xi_2^\mu \\ . \\ \xi_4^\mu \end{bmatrix}$$

y produce la salida

$$V_j^\mu = g(h_j^\mu)$$

que puede expresarse como

$$\begin{bmatrix} V_1^\mu \\ V_2^\mu \\ . \\ . \\ . \\ V_{20}^\mu \end{bmatrix} = \begin{bmatrix} g(h_1^\mu) \\ g(h_2^\mu) \\ . \\ . \\ . \\ g(h_{20}^\mu) \end{bmatrix}$$

La unidad de salida i recibe entonces

$$h_i^\mu = \sum_j W_{ij} V_j^\mu$$

que se puede expresar como

$$\begin{bmatrix} h_1^\mu \\ h_2^\mu \\ . \\ h_4^\mu \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & . & . & . & w_{1,20} \\ w_{21} & w_{22} & . & . & . & w_{2,20} \\ . & . & . & . & . & . \\ w_{41} & . & . & . & . & w_{4,20} \end{bmatrix} \begin{bmatrix} \xi_1^\mu \\ \xi_2^\mu \\ . \\ . \\ . \\ \xi_{20}^\mu \end{bmatrix}$$

y produce la salida final

$$O_i^\mu = g(h_i^\mu)$$

que puede expresarse como

$$\begin{bmatrix} O_1^\mu \\ O_2^\mu \\ . \\ O_4^\mu \end{bmatrix} = \begin{bmatrix} g(h_1^\mu) \\ g(h_2^\mu) \\ . \\ g(h_4^\mu) \end{bmatrix}$$

2.1. Función de activación

Para la función de activación $g(h)$ normalmente se utiliza una función sigmoidea. Esta función debe ser diferenciable y lo normal es desear que se sature en ambos extremos. Tenemos como ejemplo:

$$g(h) = f_\beta(h) = \frac{1}{1+\exp(-2\beta h)} \qquad \text{para rango } 0 \leq g(h) \leq 1 \tag{2.1}$$

Su derivada es

$$g'(h) = 2\beta g(1-g)$$

y

$$g(h) = \tanh \beta h \qquad \text{para rango } -1 \leq g(h) \leq 1 \tag{2.2}$$

Su derivada es

$$g'(h) = \beta(1-g^2) = \beta(1-(\tanh \beta h)^2)$$

El parámetro β a menudo es $1/2$ para la ecuación 2.1 y 1 para la ecuación 2.2.

2.2. Backpropagation

Consiste en corregir los pesos w_{ij} para cada capa a partir del error entre la salida deseada y la salida actual de la red para un patrón determinado. Cuando la actualización de los pesos se realiza luego de enseñar cada patrón se dice que el aprendizaje es "on-line", y cuando la actualización se realiza luego de enseñar todos los patrones se le llama aprendizaje por "epoch" (en inglés también se le llama "batch learning"). La propagación del error se realiza de la capa de salida hacia atrás, por ello su nombre. Los pasos para realizar las correcciones son:

1. Inicializamos los pesos a valores aleatorios pequeños, escogemos un patrón a enseñar ξ_k^μ, lo aplicamos a la capa de entrada y propagamos la señal hacia adelante hallando salidas para cada capa hasta obtener la salida de la última capa O_i^μ.

2. Hallamos un delta de error δ_i^μ para las conexiones de la capa escondida a la capa de salida como

   $$\delta_i^\mu = g'(h_i^\mu)[\zeta_i^\mu - O_i^\mu]$$

   comparando las salidas actuales O_i^μ con las salidas deseadas ζ_i^μ. Si usamos la función de activación de la ecuación 2.2 queda

   $$\delta_i^\mu = \beta(1-(\tanh \beta h_i^\mu)^2)[\zeta_i^\mu - O_i^\mu]$$

   Esto se puede escribir como

   $$\begin{bmatrix} \delta_1^\mu \\ \delta_2^\mu \\ . \\ \delta_4^\mu \end{bmatrix} = \beta \begin{bmatrix} (1-(\tanh \beta h_1^\mu)^2)[\zeta_1^\mu - O_1^\mu] \\ (1-(\tanh \beta h_2^\mu)^2)[\zeta_2^\mu - O_2^\mu] \\ . \\ (1-(\tanh \beta h_4^\mu)^2)[\zeta_4^\mu - O_4^\mu] \end{bmatrix}$$

3. Hallamos ΔW_{ij} para los pesos entre la capa escondida y la capa de salida

   $$\Delta W_{ij} = \eta \sum_\mu \delta_i^\mu V_j^\mu$$

   esto puede escribirse también como

   $$\begin{bmatrix} \Delta W_{11} & \Delta W_{12} & . & . & . & \Delta W_{1,20} \\ \Delta W_{21} & \Delta W_{22} & . & . & . & \Delta W_{2,20} \\ . & . & . & . & . & . \\ \Delta W_{41} & \Delta W_{42} & . & . & . & \Delta W_{4,20} \end{bmatrix} = \eta \begin{bmatrix} \delta_1^\mu \\ \delta_2^\mu \\ . \\ \delta_4^\mu \end{bmatrix} . \begin{bmatrix} V_1^\mu & V_2^\mu & . & . & . & V_{20}^\mu \end{bmatrix}$$

y actualizamos los pesos de la capa

$$W_{ij}^{nuevo} = W_{ij}^{viejo} + \Delta W_{ij}$$

4. Hallamos δ_j^μ para la capa siguiente

$$\delta_j^\mu = g'(h_j^\mu) \sum_i W_{ij}\delta_i^\mu$$

Esto queda

$$\delta_j^\mu = \beta(1 - (\tanh \beta h_j^\mu)^2) \sum_i W_{ij}\delta_i^\mu$$

Reescribiendo tenemos

$$\begin{bmatrix} \delta_1^\mu \\ \delta_2^\mu \\ . \\ . \\ . \\ \delta_{20}^\mu \end{bmatrix} = \beta \begin{bmatrix} (1 - (\tanh \beta h_1^\mu)^2)(W_{11}\delta_1^\mu + W_{21}\delta_2^\mu + W_{31}\delta_3^\mu + W_{41}\delta_4^\mu) \\ (1 - (\tanh \beta h_2^\mu)^2)(W_{12}\delta_1^\mu + W_{22}\delta_2^\mu + W_{32}\delta_3^\mu + W_{42}\delta_4^\mu) \\ . \\ . \\ . \\ (1 - (\tanh \beta h_{20}^\mu)^2)(W_{1,20}\delta_1^\mu + W_{2,20}\delta_2^\mu + W_{3,20}\delta_3^\mu + W_{4,20}\delta_4^\mu) \end{bmatrix}$$

5. Hallamos Δw_{jk} para la misma capa

$$\Delta w_{jk} = \eta \sum_\mu \delta_j^\mu \xi_k^\mu$$

Esto puede escribirse también como

$$\begin{bmatrix} \Delta w_{11} & \Delta w_{12} & . & . & . & \Delta w_{1,4} \\ \Delta w_{21} & \Delta w_{22} & . & . & . & \Delta w_{2,4} \\ . & . & . & . & . & . \\ \Delta w_{20,1} & \Delta w_{20,2} & . & . & . & \Delta w_{20,4} \end{bmatrix} = \eta \begin{bmatrix} \delta_1^\mu \\ \delta_2^\mu \\ . \\ \delta_{20}^\mu \end{bmatrix} \cdot \begin{bmatrix} \xi_1^\mu & \xi_2^\mu & . & . & . & \xi_4^\mu \end{bmatrix}$$

y actualizamos los pesos

$$w_{jk}^{nuevo} = w_{jk}^{viejo} + \Delta w_{jk}$$

Realizamos esto para cada patrón presentado a la red hasta presentar todos los patrones. A cada presentación de todos los patrones se le llama un "epoch". El aprendizaje se extenderá por tantos "epoch" como sea necesario hasta que se cumpla el criterio de convergencia o bien hasta un número de "epochs" determinado por el usuario de la red.

2.2.1. Online Backpropagation

Consiste en actualizar los pesos cada vez que se presenta un patrón a la red. Para los pesos de todas las capas el algoritmo es:

```
for EPOCH=1 to MAXEPOCH
    for PATRON=1 to NPATRONES
        ΔW_ij = ηδ_i V_j
        W_ij = W_ij + ΔW_ij
    end
end
```

2.2.2. Batch Backpropagation

También se le llama "learning by epoch". Consiste en actualizar los pesos cada vez que se presentan todos los patrones a la red. Para esta forma de aprendizaje no se requiere entrenar la red con los patrones seleccionados aleatoriamente, ya que el delta de peso total para todos los patrones es independiente del orden en que son enseñados. Para todos los pesos de la red el algoritmo es:

for EPOCH=1 to MAXEPOCH
    for PATRON=1 to NPATRONES
        $\Delta W_{ij} = \Delta W_{ij} + \eta \delta_i V_j$
    end
$W_{ij} = W_{ij} + \Delta W_{ij}$
end

2.3. Algoritmo RPROP

Este algoritmo fue propuesto en 1993 por Martin Riedmiller y Heinrich Braun y aplica para redes multicapas como el Perceptrón. El algoritmo realiza una adaptación local de las actualizaciones de los pesos de acuerdo al comportamiento de la función de error. El buen funcionamiento del algoritmo depende solamente del comportamiento temporal del signo de la derivada y no de su tamaño.

La actualización de los pesos en el algotimo de Backpropagation se realiza de acuerdo a

$$w_{ij}^{nuevo} = w_{ij}^{viejo} + \Delta w_{ij}$$

donde

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

η es la constante de aprendizaje que ya conocemos.

RPROP significa "resilient propagation". El algoritmo realiza una adaptación directa del paso Δw_{ij} basado en información local del gradiente. Para conseguir esto se intruduce un valor de actualización individual para cada peso, Δ_{ij}, el cual por si solo determina el tamaño de Δw_{ij}. Los valores de Δ_{ij} evolucionan durante el proceso de aprendizaje basados en su mirada local de la función de error E de acuerdo a la regla de aprendizaje siguiente:

$$\Delta_{ij}^{(t)} = \begin{cases} \rho^+ * \Delta_{ij}^{(t-1)}, & \text{si } \frac{\partial E}{\partial w_{ij}}^{(t-1)} * \frac{\partial E}{\partial w_{ij}}^{(t)} > 0 \\ \rho^- * \Delta_{ij}^{(t-1)}, & \text{si } \frac{\partial E}{\partial w_{ij}}^{(t-1)} * \frac{\partial E}{\partial w_{ij}}^{(t)} < 0 \\ \Delta_{ij}^{(t-1)}, & \text{caso faltante} \end{cases}$$

donde

$$0 < \rho^- < 1$$

$$1 < \rho^+ < \infty$$

para la teoría de la referencia [8] que hemos estudiado, el valor de la derivada es

$$\frac{\partial E}{\partial w_{ij}} = -\sum_{\mu} \delta_i^{\mu} V_j^{\mu}$$

donde δ_i^{μ} corresponde a la capa (m) y V_j^{μ} a la capa $(m-1)$. Los deltas de pesos se hallan como

$$\Delta w_{ij}^{(t)} = \begin{cases} -\Delta_{ij}^{(t)}, & \text{si } \frac{\partial E}{\partial w_{ij}}^{(t)} > 0 \\ +\Delta_{ij}^{(t)}, & \text{si } \frac{\partial E}{\partial w_{ij}}^{(t)} < 0 \\ 0, & \text{caso faltante} \end{cases}$$

La actualización de los pesos se daría como

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + \Delta w_{ij}^{(t)}$$

Sin embargo, existe una excepción: si la derivada parcial cambia de signo, esto es, si el paso previo fue demasiado largo y nos saltamos el mínimo, debemos revertir la última actualización de ese peso, así:

$$\Delta w_{ij}^{(t)} = -\Delta w_{ij}^{(t-1)}, \quad \text{si } \frac{\partial E}{\partial w_{ij}}^{(t-1)} * \frac{\partial E}{\partial w_{ij}}^{(t)} < 0$$

Capítulo 3

Relación Constitutiva Elástica Lineal

3.1. Caso general — 3D

Se considerará el caso más simple para una relación tensión-deformación, donde la relación constitutiva es lineal debido a que está determinada por números constantes, e isótropa porque el material tiene el mismo comportamiento en todas las direcciones y está determinado sólo por dos constantes, el módulo de Young E y el coeficiente de poisson ν. El modelo clásico de la relación constitutiva lineal entre tensiones y deformaciones para el caso más general (3D) está dado por la ecuación 3.1:

$$\begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \tau_{12} \\ \tau_{23} \\ \tau_{13} \end{bmatrix} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1 & \frac{\nu}{1-\nu} & \frac{\nu}{1-\nu} & 0 & 0 & 0 \\ \frac{\nu}{1-\nu} & 1 & \frac{\nu}{1-\nu} & 0 & 0 & 0 \\ \frac{\nu}{1-\nu} & \frac{\nu}{1-\nu} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} \end{bmatrix} \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ \gamma_{12} \\ \gamma_{23} \\ \gamma_{13} \end{bmatrix} \tag{3.1}$$

3.2. Deformación plana — 2D

Cuando se desean analizar estructuras que poseen una dimensión mucho mayor que las otras dos nos encontramos frente a un caso de deformación plana. En este trabajo nos interesa estudiarla debido a que los datos de entrenamiento de la red pueden ser extraídos de códigos de Matlab cortos que no requieran demasiado tiempo de desarrollo. Los procesos de laminado, extruido y forjado son modelados como casos de deformación plana, donde las deformaciones están sobre un plano 2D, más no las tensiones. Esto indica que:

$$\epsilon_{33} = \epsilon_{13} = \epsilon_{23} = 0$$

$$\sigma_{13} = \sigma_{23} = 0$$

Si tenemos en cuenta el modelo 3D y aplicamos las condiciones anteriores obtendremos que el modelo constitutivo matricial para éste caso sería entonces el siguiente:

$$\begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \tau_{12} \\ 0 \\ 0 \end{bmatrix} = [C] \begin{bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ 0 \\ \gamma_{12} \\ 0 \\ 0 \end{bmatrix}$$

Simplificando la expresión anterior obtendríamos:

$$\begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \tau_{12} \end{bmatrix} = \frac{E}{(1+v)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & 0 \\ \nu & 1-\nu & 0 \\ 0 & 0 & \frac{(1-2\nu)}{2} \end{bmatrix} \begin{bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ 2\epsilon_{12} \end{bmatrix} \tag{3.2}$$

Donde $\gamma_{12} = 2\epsilon_{12}$. De $\sigma = C\epsilon$ (3D) obtengo:

$$\sigma_{33} = \frac{E\nu}{(1+\nu)(1-2\nu)}(\epsilon_{11} + \epsilon_{22}) + \frac{E(1-\nu)}{(1+\nu)(1-2\nu)}\epsilon_{33}$$

Como estamos ante un caso de deformación plana la componente 33 (normal al plano) de la deformación es cero, esto es, $\epsilon_{33} = 0$, luego:

$$\sigma_{33} = \frac{E\nu}{(1+\nu)(1-2\nu)}(\epsilon_{11} + \epsilon_{22})$$

La relación constitutiva de la Ecuación 3.2 la emplearemos más adelante para entrenar una red y comparar los resultados del entrenamiento con la evaluación exacta de la relación constitutiva.

Capítulo 4

Notas sobre el entrenamiento del Perceptrón

Para entrenar la red se modifican los pesos de las interconexiones mediante un método iterativo para obtener una señal de salida lo más próxima posible a la salida deseada como respuesta a un patrón de entrada. A este proceso se le llama entrenamiento. Para ello deben conocerse de antemano datos de entrada y salida de la red. Una parte de estos datos puede ser utilizada para entrenarla, mientras que la otra parte puede usarse para realizar pruebas que confirmen un correcto aprendizaje [9].

4.1. Entrenamiento de una relación constitutiva

En el caso de que tengamos que entrenar una relación constitutiva (o ley constitutiva), el modelo no simbólico (lo que queremos hallar con la monografía) es contruído como sigue: primero, se entrena una red con un grupo de datos de entrada y salida, el cual puede obtenerse a partir de pruebas numéricas o experimentos. Una vez finaliza el entrenamiento, la capacidad de generalización de la red nos permite predecir el comportamiento del material, por ejemplo, producir la relación tensión-deformación (C) para una secuencia arbitraria de valores de tensiones y deformaciones. De ésta forma, el procedimiento para modelar relaciones constitutivas se divide en tres pasos importantes:

1. Se selecciona la topología de la Red Neuronal Artificial (RNA) que pueda representar el problema. El artículo de la Referencia [9] limita esta opción sólo a redes con capas escondidas, sin embargo el artículo de la Referencia [4] logra emplear una red sin capas escondidas para modelar una relación constitutiva elástica lineal.

2. Adquisición de datos que contengan la información que caracteriza el problema muestreando el comportamiento de un material mediante modelación numérica o experimentación real.

3. Inclusión de la RNA como una subrutina que describe el comportamiento del material en un código (de elementos finitos, por ejemplo).

Está comprobado que las RNA son aproximadores universales para cualquier función con muchas variables independientes [1]. Para este caso, las variables independientes corresponden con las deformaciones. La información sobre la dependencia funcional es transferida a la RNA por medio de un número de datos discretos que describan el valor

de la función para un número suficientemente largo de puntos de muestra en el espacio de las variables de interés.

Desafortunadamente, los teoremas que aseguran que las RNA pueden aproximar cualquier función de muchas variables no pueden construirse. Esto significa que la cantidad correcta de grados de libertad de éste operador (por ejemplo, el número de capas escondidas, el número de nodos o neuronas, la repartición de pesos entre capas escondidas que determinan la cantidad de pesos y bias) tienen que ser definidas por el usuario. Estas características son dependientes del problema y no es posible dar una prescripción general, por lo cual la práctica común es determinar la mejor configuración topológica tratando de minimizar el número de capas ocultas y el número de neuronas por cada capa [1].

4.1.1. Posibilidades de entrenamiento

Las RNA con múltiples nodos en la salida pueden ser reemplazadas por grupos de RNA donde cada uno tenga una sola salida, y donde todos los grupos sean entrenados con los mismos patrones. Se ha notado que a veces de ésta forma el proceso de entrenamiento da mejores resultados [1].

4.1.2. Normalización de datos

Antes de entrenar la red tanto las variables de entrada como las de salida deberán ser normalizadas en cierto rango. Dependiendo de la función de activación que utilicemos podemos necesitar normalizar en un rango de 0 a 1, tal como recomiendan las referencias [14] y [10]. Esto nos llevaría a usar un método de normalización ampliamente utilizado, que es

$$X' = \frac{X - 0{,}95X_{min}}{1{,}05X_{max} - 0{,}95X_{min}} \tag{4.1}$$

donde X son los datos originales, y X_{min} y X_{max} son los valores mínimos y máximos de X, respectivamente. X' son los datos unificados de los correspondientes X. Este procedimiento de pre-procesamiento puede hacer más eficiente el entrenamiento de la red neuronal [13].

Otra forma de normalizar los datos de entrada es la propuesta por Hashash, Jung y Ghaboussi en la Referencia [7]. Se emplea cuando los datos corresponden a deformaciones que serán empleadas en la capa de entrada de la red neuronal. El vector de deformaciones sería escalado entonces por un factor apropiado:

$$\varepsilon_i^{NN} = \frac{\varepsilon_i}{S_i^{\varepsilon}} \quad \text{tal que} \quad -1 < \varepsilon_i^{NN} < 1 \tag{4.2}$$

Los datos de salida también deben ser escalados de vuelta. Siguiendo la notación de la Referencia [7] tendríamos:

$$\sigma_i = S_i^{\sigma}\sigma_i^{NN} \quad \text{donde} \quad -1 < \sigma_i^{NN} < 1$$

4.1.3. Criterio de Convergencia

El criterio de convergencia para la red está determinado por el promedio de la raiz media cuadrática del error (RMS por sus siglas en inglés) entre los valores deseados y los de salida, y está dado como,

$$E_{RMS} = \frac{1}{N}\sum_{i=1}^{N}\sqrt{\frac{1}{p}\sum_{j=1}^{p}(d_{ij} - y_{ji})^2} \tag{4.3}$$

donde E_{RMS} es el RMS promedio, N es el número de datos de entrenamiento o prueba, p es el número de variables en la salida, $d_j(n)$ y $y_j(n)$ son la salida objetivo y la salida de la red para la neurona j, respectivamente [10]. Este criterio fue el elegido para este trabajo, de modo que el proceso iterativo de aprendizaje será detenido cuando el error permitido sea inferior a E_{RMS}.

4.1.4. Cantidad óptima de neuronas en capas ocultas

El número óptimo de neuronas debe reducirse al mínimo para prevenir la tendencia que tiene la red a sobre-ajustarse a los datos [3]. Es posible implementar un algoritmo de adaptación del número de neuronas, el cual inicia con pocas neuronas y las va incrementando a medida que se requiera un mejor ajuste de los pesos a los patrones de entrada y salida. Ver referencia [6] para más información al respecto.

4.1.5. Backpropagation: Aumento de velocidad de convergencia

Para mejorar este aspecto del entrenamiento del perceptrón se utilizará la introducción de un término de momentum para el cálculo del $\Delta w_{pq}(t+1)$. La idea es darle a cada conexión w_{pq} una inercia o momenum de forma que tienda a cambiar en la dirección de descenso que sienta más "fuerza", en lugar de oscilar con pequeñas variaciones. Este esquema se implementa dándole a cada $\Delta w_{pq}(t+1)$ una contribución de peso proveniente del paso de tiempo anterior:

$$\Delta w_{pq}(t+1) = -\eta\frac{\partial E}{\partial w_{pq}} + \alpha\Delta w_{pq}(t) \tag{4.4}$$

El parámetro de momentum α debe estar entre 0 y 1. A menudo se utiliza un valor de 0.9 [8].

Capítulo 5

Ejercicio 1: Caso Elástico Lineal 1

El objetivo principal de éste y de los siguientes dos capítulos es comprobar la posibilidad de desarrollar un modelo constitutivo a partir de datos obtenidos mediante mediciones experimentales. En lugar de emplear resultados de pruebas reales se generarán datos sintéticos, es decir, números generados por computadora para el entrenamiento. Esto debido a la falta de datos experimentales reales.

5.1. Entrenamiento con todas las componentes del vector de deformaciones y tensiones

Para este ejercicio se logró entrenar una relación constitutiva elástica lineal con datos de deformaciones y tensiones que fueron obtenidos con deformaciones en un rango entre $-0{,}0064$ y $+0{,}0064$. Se creó una matriz de datos con 64 estados de deformación (patrones de entrada) con los cuales se entrenó la red. Las componentes de cada vector patrón son ε_{11}, ε_{22} y γ_{12}. Cada patrón se obtuvo evaluando la siguiente relación:

$$\begin{bmatrix}\sigma_{11}\\ \sigma_{22}\\ \tau_{12}\end{bmatrix} = \frac{E}{(1+v)(1-2\nu)}\begin{bmatrix}1-\nu & \nu & 0\\ \nu & 1-\nu & 0\\ 0 & 0 & \frac{(1-2\nu)}{2}\end{bmatrix}\begin{bmatrix}\epsilon_{11}\\ \epsilon_{22}\\ \gamma_{12}\end{bmatrix} \tag{5.1}$$

$$\sigma_{33} = \frac{E\nu}{(1+\nu)(1-2\nu)}(\epsilon_{11}+\epsilon_{22}) \tag{5.2}$$

que se puede resumir como

$$\begin{bmatrix}\sigma_{11}\\ \sigma_{22}\\ \tau_{12}\\ \sigma_{33}\end{bmatrix} = \frac{E}{(1+v)(1-2\nu)}\begin{bmatrix}1-\nu & \nu & 0\\ \nu & 1-\nu & 0\\ 0 & 0 & \frac{(1-2\nu)}{2}\\ \nu & \nu & 0\end{bmatrix}\begin{bmatrix}\epsilon_{11}\\ \epsilon_{22}\\ \gamma_{12}\end{bmatrix} \tag{5.3}$$

Si evaluamos $E = 2{,}1e+6$ y $\nu = 0{,}3$ en la ecuación 5.3 obtendremos la relación que deberíamos encontrar luego de entrenar la red:

$$C = \begin{bmatrix}2{,}8269e+06 & 1{,}2115e+06 & 0\\ 1{,}2115e+06 & 2{,}8269e+06 & 0\\ 0 & 0 & 8{,}0769e+05\\ 1{,}2115e+06 & 1{,}2115e+06 & 0\end{bmatrix} \tag{5.4}$$

La topología de la red es similar a la presentada en la figura 5.1, donde se posee un Perceptrón simple con 4 entradas y 4 salidas. La cuarta entrada es un valor fijo igual a 1 (umbral).

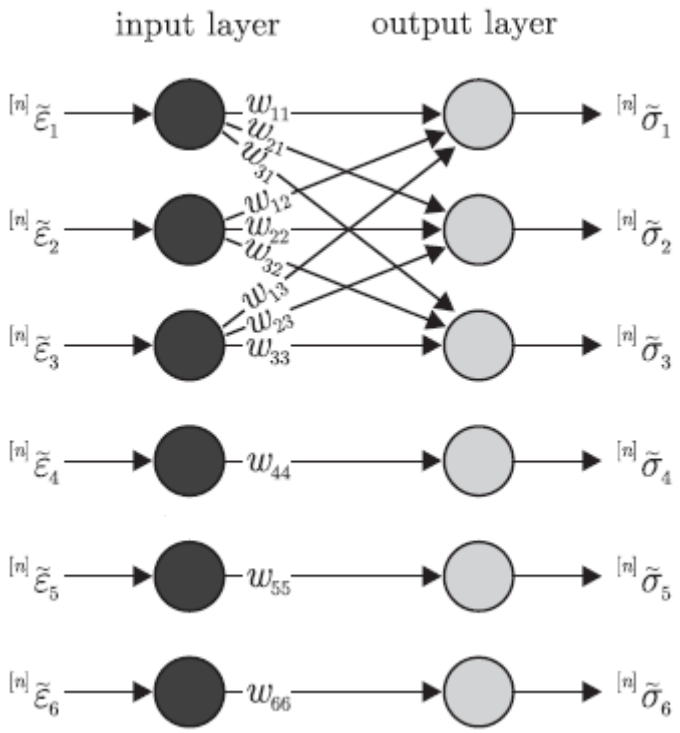

Figura 5.1: Red alimentada hacia adelante para modelo de material elástico lineal. Tomada de [4].

La labor del Perceptrón implementado es, entonces, lograr hallar las componentes de la relación constitutiva matricial, es decir, los pesos sinápticos de la red. Las entradas fueron escaladas a un rango entre -1 y 1 al igual que las salidas. Luego, el resultado de la matriz de pesos es escalada para poder obtener las componentes deseadas. Para ello debemos tener en cuenta la relación constitutiva a escala completa producto del aprendizaje C_{nn}.

$$\sigma = C_{nn}\varepsilon \tag{5.5}$$

Ahora tenemos en cuenta la relación a escala, es decir nuestra red neuronal, donde σ' es el vector de salidas, ε' es el vector de entradas y $C'_{nn} = W$ es la matriz de pesos entre las dos capas.

$$\sigma' = C'_{nn}\varepsilon' = W\varepsilon' \tag{5.6}$$

Siguiendo la convención para los subíndices de los pesos hallada en los textos, y apreciada en la figura 5.1 tenemos que

$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \\ w_{41} & w_{42} & w_{43} & w_{44} \end{bmatrix} \tag{5.7}$$

Ahora tenemos en cuenta las escalas de las entradas y las salidas

$$\sigma' = A_\sigma \sigma \quad y \quad \varepsilon' = A_\varepsilon \varepsilon \tag{5.8}$$

donde A_{sigma} es el número que escala las tensiones y A_ε el que escala las deformaciones, ambas, a un rango entre -1 y 1. Reemplazando en la ecuación 5.6 y reordenando obtenemos

$$\{\sigma\} = \left[\frac{A_\varepsilon}{A_\sigma} C'_{nn}\right] \{\varepsilon\} \tag{5.9}$$

Por igualdad con la ecuación 5.5 finalmente obtenemos la ecuación para escalar los pesos y transformarlos a componentes del tensor constitutivo

$$C_{nn} = \left[\frac{A_\varepsilon}{A_\sigma} C'_{nn}\right] = \left[\frac{A_\varepsilon}{A_\sigma} W\right] \tag{5.10}$$

5.2. Función de activación lineal

Para lograr el entrenamiento exitoso de la red se empleó una red sin capas ocultas y una función de activación lineal para modelar el comportamiento de un material elástico lineal, tal como lo sugiere la referencia [4], esto es

$$g(h) = h$$

su derivada sería entonces

$$g'(h) = 1$$

La referencia [4] también sugiere que algunos pesos sean igualados a cero, pero los resultados numéricos hallados sugieren que esto no es estrictamente necesario y por ello no se implementó.

5.3. Escalamiento de datos de entrada y salida

El escalamiento de los datos de entrada y salida se realizó de acuerdo a lo propuesto por la ecuación 4.2. Las entradas fueron escaladas en un rango entre -1 y 1, al igual que las salidas. Para esto, todos los valores fueron divididos sobre el dato con mayor valor absoluto.

5.4. Estados enseñados a la red

En la figura 5.2 se aprecia una gráfica, donde cada punto representa un estado de tensiones y deformaciones. Debido a que un estado está representado por tres deformaciones y cuatro tensiones es necesario graficar las deformaciones y tensiones equivalentes. Las fórmulas para el caso más general (3D), son:

$$\epsilon_{eq} = \frac{\sqrt{2}}{3}\left[(\epsilon_x - \epsilon_y)^2 + (\epsilon_y - \epsilon_z)^2 + (\epsilon_z - \epsilon_x)^2 + 6(\gamma_{xy}^2 + \gamma_{yz}^2 + \gamma_{zx}^2)\right]^{1/2} \tag{5.11}$$

$$\sigma_{eq} = \frac{1}{\sqrt{2}}\left[(\sigma_x - \sigma_y)^2 + (\sigma_y - \sigma_z)^2 + (\sigma_z - \sigma_x)^2 + 6(\tau_{xy}^2 + \tau_{yz}^2 + \tau_{zx}^2)\right]^{1/2} \tag{5.12}$$

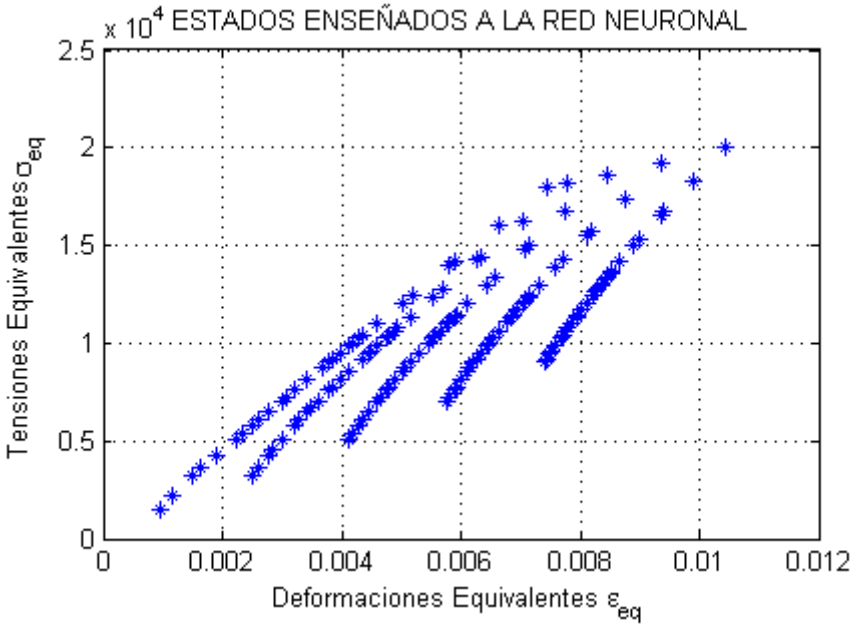

Figura 5.2: Estados enseñados a la red para modelo de material elástico lineal e isótropo. Campo de estados para 1000 parejas de patrones enseñadas.

5.5. Resultados

En la tabla 5.1 se resumen los parámetros de simulación empleados para obtener la relación constitutiva de un material elástico, lineal e isótropo a partir de datos de deformaciones (entradas) y tensiones (salidas). Al algoritmo de Backpropagation se le adicionó la teoría del momentum para el cálculo de los deltas de pesos, como se estudió en la sección 4.1.5.

Parámetro	**Valor**
Número de Patrones	64
Constante de aprendizaje η	0.001
Constante de momentum α	0.9
Error RMS mínimo $E_{rms,min}$	1e-10
Convergencia: Iteración, E_{rms}	60, 9.71e-011
Rango de deformaciones de entrenamiento	$-0{,}0064 \leq \varepsilon \leq +0{,}0064$

Cuadro 5.1: Parámetros de simulación para modelo elástico lineal isótropo.

La relación constitutiva encontrada, luego de escalar los pesos de acuerdo a la ecuación 5.10 es:

$$C_{nn} = \begin{bmatrix} 2{,}8269e+06 & 1{,}2115e+06 & \to 0 & \to 0 \\ 1{,}2115e+06 & 2{,}8269e+06 & \to 0 & \to 0 \\ \to 0 & \to 0 & 8{,}0769e+05 & \to 0 \\ 1{,}2115e+06 & 1{,}2115e+06 & \to 0 & \to 0 \end{bmatrix}$$

$\to 0$ indica que los pesos en esas posiciones tienden a cero conforme se disminuye el $E_{rms,min}$ permisible.

5.6. Conclusión

Si comparamos los números del modelo de material C_{nn} con los números del modelo original C de la ecuación 5.4 podemos constatar que el aprendizaje de la relación constitutiva elástica lineal isótropa a partir de datos de tensiones y deformaciones se efectuó de forma exitosa ya que se logró estimar los valores de las componentes de la matriz sólo a partir de datos de deformaciones y tensiones.

No se considera necesario realizar pruebas del modelo elástico lineal, ya que las componentes encontradas de la matriz que representa el modelo del material reproducen con buena exactitud a las componentes del modelo clásico, y el modelo de red simple hace coincidir cada componente con un solo peso. Por esto se puede concluir directamente que el comportamiento será el mismo, as el campo de las deformaciones de prueba esté por fuera del campo de las deformaciones con las que se entrenó la red. En los próximos ejercicios pondremos a prueba la red para casos de aprendizaje más complejos que éste.

Como otro aporte que puede resultar interesante se destaca que obtener la matriz constitutiva permite también hallar las dos contantes del material, por ahora, sólo para el modelo elástico lineal. Las constantes que se puede hallar son el módulo elástico o de Young E y el coeficiente de Poisson ν.

Capítulo 6

Ejercicio 2: Caso Elástico Lineal 2

6.1. Entrenamiento sólo con componentes principales (aplicable a la realidad)

En la sección 5.1 se entrenó una red disponiendo de datos de deformaciones y tensiones normales y cortantes. En ésta sección, y para éste ejercicio, tomaremos sólo las componentes normales, debido a que los ensayos triaxiales reales no producen datos de deformación ni tensión cortantes por lo complejo de su medición. Para entrenar la red se utilizará la técnica de enriquecimiento de datos propuesta por Shin & Pande (2002) en la Referencia [12] para obtener un buen desempeño del modelo constitutivo modelado con la red reuronal (en inglés, Neural Network Constitutive Model o NNCM).

6.2. Sobre los ensayos triaxiales

Una posibilidad para la obtención de los datos con los cuales se podría entrenar la red proviene de ensayos triaxiales sobre una muestra del material bajo estudio, la cual es sometida a ciertas configuraciones de carga que garantizan la uniformidad del campo de tensiones y deformaciones. Una vez los datos de tensiones y deformaciones de los ensayos triaxiales son capturados, se utilizan para entrenar la red que modela la relación constitutiva. Las mediciones habrán sido tomadas de mediciones sobre el volumen finito de la muestra, pero pueden ser empleadas para entrenar la relación aunque ésta esté definida para relacionar estados que están definidos en un sólo punto del material. Esto debido a los campos uniformes de tensiones y deformaciones que se logran durante los experimentos y que mencionamos al principio del párrafo.

6.3. Datos de entrenamiento de la red

En lugar de usar datos experimentales reales para el material se usarán datos generados (también reciben el nombre de 'sintéticos') a partir de la ya conocida relación constitutiva matricial. Para éste propósito se emplearon parámetros típicos usados para modelar metales con una relación constitutiva isotrópica. Definimos el módulo de Young $E = 2{,}1e6\ Pa$ y el coeficiente de Poisson $\nu = 0{,}3$. Los patrones de tensión y deformación serán generados de acuerdo a lo sugerido por la Referencia [2], donde usaremos datos de tensiones y deformaciones para condiciones de carga a tensión y compresión para cada

uno de los dos ejes principales y serán complementados con la técnica de enriquecimiento de datos vista en la sección 6.4.

Queremos generar datos sin términos cortantes, tal como son obtenidos los datos experimentales de las pruebas triaxiales, y comprobar, utilizando la técnica de enriquecimiento de datos, que es posible obtener de nuevo la relación constitutiva elástica lineal e isótropa. Los datos de deformaciones principales serán obtenidos mediante la siguiente relación:

$$\begin{bmatrix}\sigma_1\\ \sigma_2\\ \sigma_3\end{bmatrix} = \frac{E}{(1+v)(1-2\nu)}\begin{bmatrix}1-\nu & \nu\\ \nu & 1-\nu\\ \nu & \nu\end{bmatrix}\begin{bmatrix}\epsilon_1\\ \epsilon_2\end{bmatrix} \tag{6.1}$$

Donde σ_1 y σ_2 son las tensiones principales y ε_1 y ε_2 son las deformaciones principales. La única diferencia entre este ejercicio y el anterior son los datos de entrenamiento de la red. En la sección 6.4 se describe la técnica de enriquecimiento de datos, para aprovechar las componentes principales y generar con ellas un set de patrones con más información disponible para entrenar la red.

6.4. Enriquecimiento de datos

Las parejas de vectores de tensiones y deformaciones que provienen de las pruebas triaxiales son en realidad tensiones y deformaciones principales. Esto significa que son mediciones tomadas a lo largo de ejes ortogonales entre sí, lo que en términos de campos de tensiones y deformaciones implica que no existen componentes cortantes. Si utilizáramos las componentes principales para entrenar la red ésta tendría que extrapolar los datos cortantes, debido a que no fué entrenada con parejas de vectores que contuvieran esta información. Esto conduciría a grandes imprecisiones en la respuesta tensión-deformación de la red.

Para superar esta limitante, Shin & Pande [12] propusieron una estrategia de enriquecimiento de datos para crear parejas de vectores (patrones) transformando tensiones y deformaciones a valores donde las componentes cortantes sean diferentes de cero. Para el caso bidimensional, la transformación de un vector de deformaciones principales rotado un ángulo θ con respecto al eje X es como sigue:

$$\begin{aligned}\varepsilon_x &= C + D\cos(2\theta)\\ \varepsilon_y &= C - D\cos(2\theta)\\ \gamma_{xy} &= 2D\sin(2\theta)\end{aligned}$$

donde

$$\begin{aligned}C &= \frac{1}{2}(\varepsilon_1 + \varepsilon_2)\\ D &= \frac{1}{2}(\varepsilon_1 - \varepsilon_2)\end{aligned}$$

Para las tensiones:

$$\sigma_x = A + B\cos(2\theta)$$
$$\sigma_y = A - B\cos(2\theta)$$
$$\tau_{xy} = B\sin(2\theta)$$

donde

$$A = \frac{1}{2}(\sigma_1 + \sigma_2)$$
$$B = \frac{1}{2}(\sigma_1 - \sigma_2)$$

Este método produce una gran cantidad de datos dependiendo del número de transformaciones elegidas para generarlos. Entre los datos expandidos hay muchas parejas de vectores repetidas de modo que se necesitan procesar previamente los datos para filtrarlas. Para el entrenamiento de la red se empleará un ángulo incremental $\Delta\theta = 15°$ para rotar los ejes de tensión-deformación desde $-45°$ hasta $+45°$.

6.5. Estados enseñados a la red

En la figura 6.1 se aprecia una gráfica, donde cada punto representa un estado de tensiones y deformaciones. Debido a que un estado está representado por tres deformaciones y cuatro tensiones es necesario graficar las deformaciones y tensiones equivalentes. Ver ecuaciones 5.11 y 5.12. Los puntos graficados en la figura 6.1 corresponden a estados de deformación y tensión luego del enriquecimiento de datos.

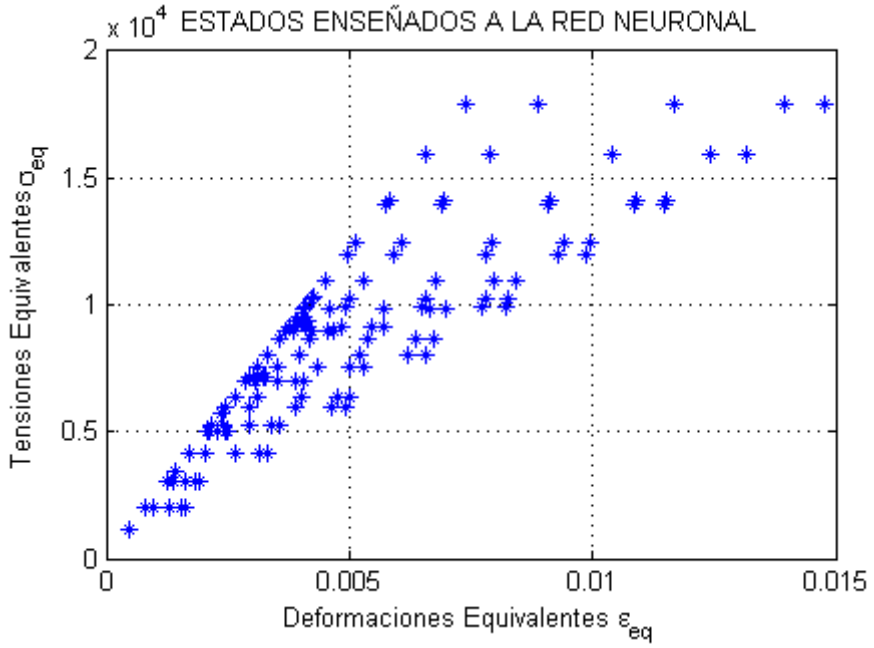

Figura 6.1: Campo de estados para 730 parejas de patrones.

6.6. Resultados

En la tabla 7.1 se resumen los parámetros de simulación empleados para obtener la relación constitutiva elástica lineal de este ejercicio. Los resultados fueron obtenidos entrenando sólo con 76 patrones, los cuales fueron obtenidos luego del enriquecimiento y del filtrado de los patrones repetidos, y partiendo de tan solo 16 parejas de vectores de deformaciones y tensiones principales.

Parámetro	**Valor**
Número de Patrones enseñados	76
Constante de aprendizaje η	0.001
Constante de momentum α	0.9
Error RMS mínimo $E_{rms,min}$	1e-10
Convergencia: Iteración, E_{rms}	314, 9.50e-011
Rango de deformaciones de entrenamiento	$-0{,}0064 \leq \varepsilon \leq +0{,}0064$

Cuadro 6.1: Parámetros de simulación para modelo elástico lineal isótropo.

La relación constitutiva encontrada, luego de escalar los pesos de acuerdo a la ecuación 5.10 es:

$$C_{nn} = \begin{bmatrix} 2{,}8269e+06 & 1{,}2115e+06 & \to 0 & \to 0 \\ 1{,}2115e+06 & 2{,}8269e+06 & \to 0 & \to 0 \\ \to 0 & \to 0 & 8{,}0769e+05 & \to 0 \\ 1{,}2115e+06 & 1{,}2115e+06 & \to 0 & \to 0 \end{bmatrix}$$

$\to 0$ indica que los pesos en esas posiciones tienden a cero conforme se disminuye el $E_{rms,min}$ permisible.

6.7. Conclusión

Se logró obtener la relación constitutiva a partir de parejas de patrones de deformación y tensión principales. Esto demuestra que a partir de datos de pruebas triaxiales reales y utilizando la técnica de enriquecimiento de datos usada por Shin & Pande en la referencia [12] es posible obtener las componentes de la matriz constitutiva para emplearse en modelos que emplean materiales elásticos lineales e isótropos.

Capítulo 7

Ejercicio 3: Caso Elásto-Plástico No Lineal

El objetivo de éste ejercicio es entrenar un perceptrón de dos capas ocultas a partir de deformaciones y tensiones obtenidas de datos sintéticos a partir de un programa de Matlab que calcula deformaciones y tensiones de un elemento finito sometido a cargas, ver figura 7.1. El material considerado es no lineal debido a que sufre deformación plástica cuando la tensión de Von Mises supera la tensión de fluencia. Cuando esto sucede, el material sufre endurecimiento por deformación, lo que hace depender el estado actual del historial de deformaciones y tensiones en ese punto. Para modelar este tipo de problemas numéricamente se requiere de métodos iterativos para satisfacer las ecuaciones del modelo en cada paso de aplicación de la carga.

7.1. Entrenamiento con componentes principales (aplicable a la realidad)

El objetivo más importante de este ejercicio es obtener unos pesos que permitan estimar tensiones a partir de deformaciones, teniendo como datos de entrada y salida sólo deformaciones y tensiones principales, que pueden ser obtenidas de ensayos reales. Se logrará el objetivo si se logran pesos que permitan buenas estimaciones al someter al elemento finito mencionado a configuraciones de carga diferentes a las configuraciones con las cuales se entrenó la red.

7.2. Datos de entrenamiento de la red

Los pasos para la obtención de los datos de entrenamiento es como sigue:

1. Se somete un elemento finito para modelar deformación plana elasto-plástica de 4 nodos a configuraciones incrementales de carga a tensión, combinando 10 incrementos de carga horizontales con 10 incrementos verticales. En la figura 7.1 se aprecian tres estados principales de carga para el entrenamiento.

2. Se calculan deformaciones y tensiones principales para cada combinación de cargas. Esto dará 3 deformaciones y 4 tensiones, igual que para los ejercicios 1 y 2 estudiados en los capítulos anteriores.

3. Se realiza un enriquecimiento de datos, como sugieren Drakos & Pande en la referencia [2].
4. El punto anterior generará algunos patrones repetidos, los cuales son eliminados.
5. Se escalan los datos de entrada y salida, como se vió en la sección 4.1.2.

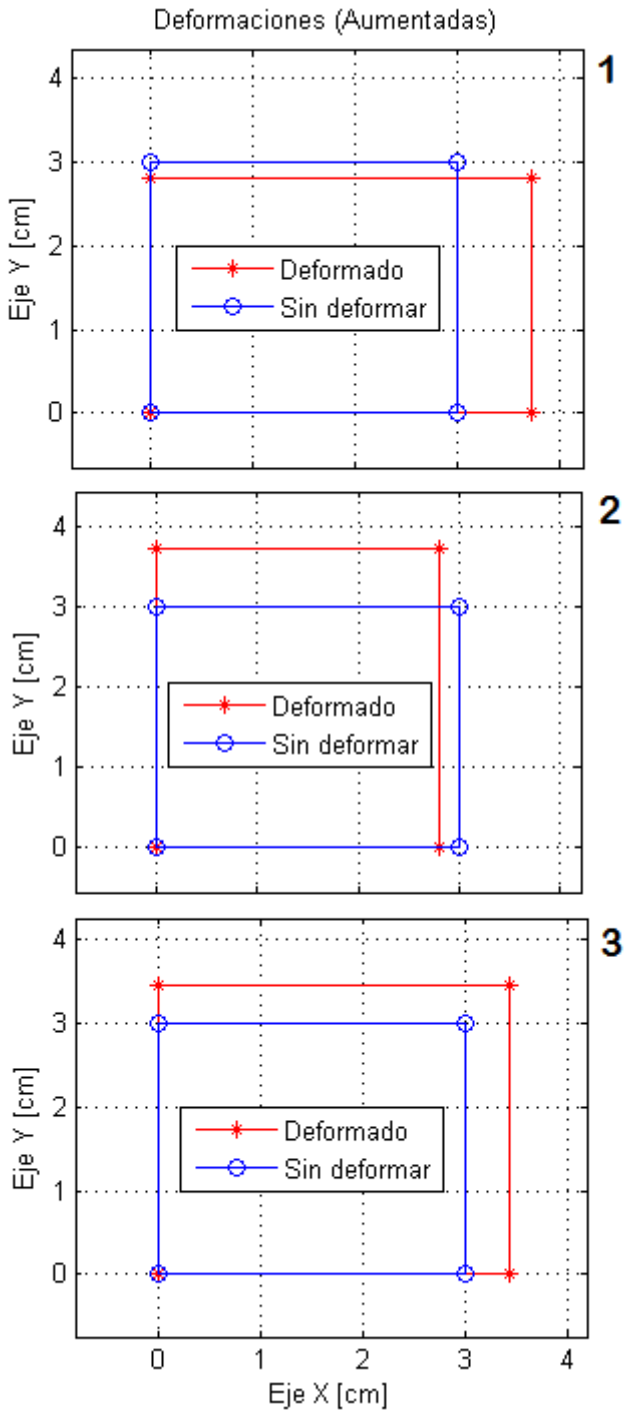

Figura 7.1: 3 de 120 estados de carga para entrenamiento de RNA. Los puntos dentro del elemento poseen estados de deformación y tensión uniformes.

Los estados de deformaciones y tensiones están compuestos por 3 y 4 componentes, respectivamente. Para facilitar la visualización de dichos estados los transformaremos a deformaciones y tensiones equivalentes, igual que como vimos en la sección 5.4.

7.2.1. Curva de carga: material elasto-plástico bilineal

En la figura 7.2 se aprecian los estados de entrenamiento sin enriquecimiento, es decir, aún no aptos para emplearse en el entrenamiento. En ella se aprecia una zona elástica, antes para $0 \leq \sigma_{eq} \leq 2500$ y una plástica para $2500 \leq \sigma_{eq} \leq \infty$. El endurecimiento por deformación se produce para estados dentro de la zona plástica.

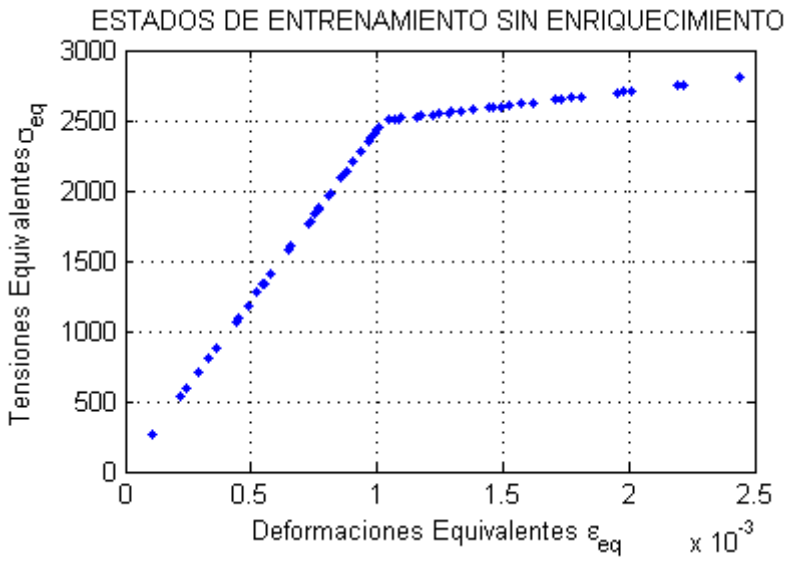

Figura 7.2: Estados de entrenamiento antes del enriquecimiento para 120 parejas de patrones. Material elasto-plástico bajo varias configuraciones de carga.

7.2.2. Enriquecimiento de datos

Esta generación adicional de datos es importante, ya que es la que permitirá poder estimar las tensiones de corte cuando se posean deformaciones con componentes cortantes. Este enriquecimiento aumenta los patrones de entrenamiento de 120 a 676, y se realiza de igual forma que para el Ejercicio 2 en la sección 6.4.

En la figura 7.3 se observan los estados de entrenamiento de la red después de haber efectuado el enriquecimiento de datos.

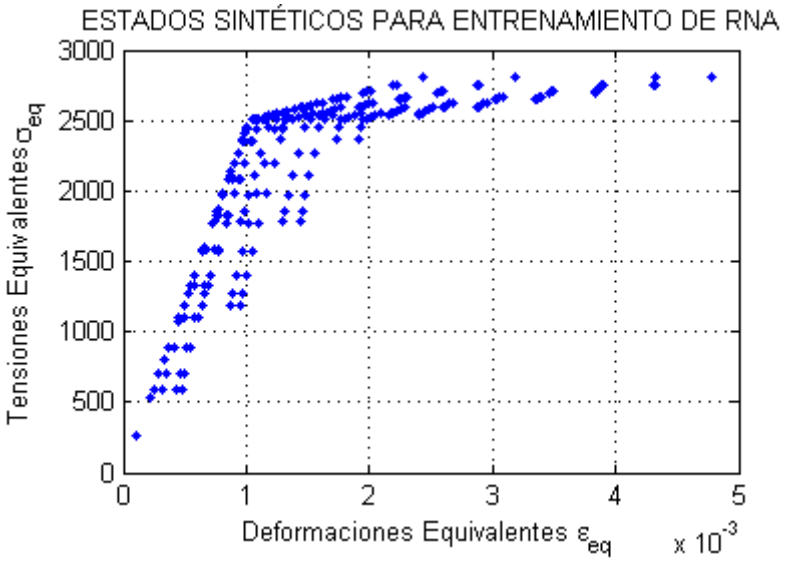

Figura 7.3: Estados de entrenamiento después del enriquecimiento para 676 parejas de patrones.

7.3. Topología de la red

Luego de numerosos intentos por obtener el mejor aprendizaje posible, se pude identificar que la topología [4-12-8-4] es la que da mejores resultados por aproximar en menor tiempo y con mayor exactitud los patrones deseados, con 4 neuronas de entrada (1 es bias) y 4 neuronas de salida, y con 2 capas ocultas de 12 y 8 neuronas cada una. La selección de ésta topología sigue aproximadamente la configuración que muestran Drakos & Pande en la página 652 de la referencia [2].

7.4. Resultado del entrenamiento

El objetivo es lograr estados aproximados por la red lo más similares posibles a los estados de entrenamiento, inicialmente. Para este ejercicio se disminuyó la constante de aprendizaje η cada 1500 iteraciones (epochs), partiendo de $\eta = 0{,}05$ y disminuyendola por 0.75 cada vez. Se utilizó la adición del términos de momentum para el cálculo de Δw_{ij} visto en la sección 4.1.5. En la tabla 7.4 se halla un resumen de los parámetros empleados, mientras que en la figura **??** se aprecian los estados logrados por la red luego del proceso de aprendizaje por Backpropagation (en rojo).

Parámetro	Valor
Número de Patrones enseñados	676
Constante de aprendizaje	$\eta < 0{,}05$
Constante de momentum α	0.9
Convergencia: Iteración, E_{rms}	30000, 4.35e-03

Cuadro 7.1: Parámetros de simulación para modelo elástico lineal isótropo.

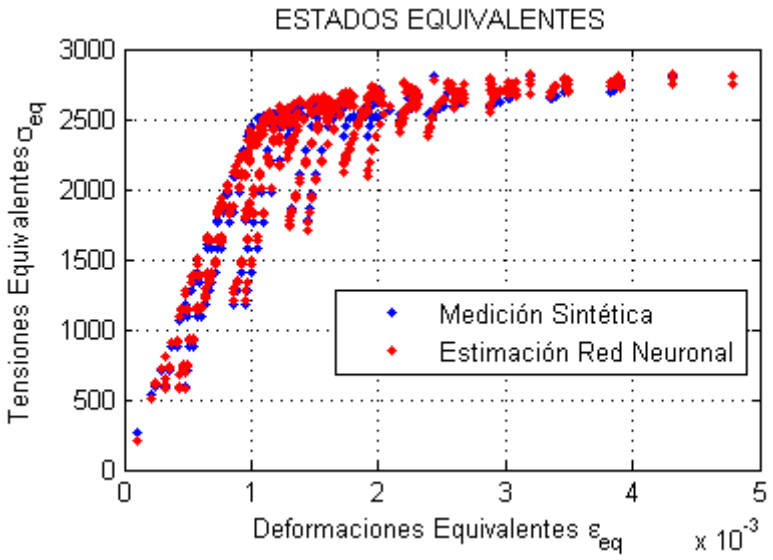

Figura 7.4: Superposición de estados de entrenamiento y estados aproximados por la red luego del aprendizaje.

7.5. Pruebas

Se requieren pruebas para constatar que el modelo constitutivo aprendido por la RNA es útil para estimar otros estados de tensión diferentes a los usados para el entrenamiento. Para ello se emplearán datos obtenidos de someter al elemento finito para deformación plana a configuraciones de carga diferentes a las usadas para obtener los datos de entrenamiento.

7.5.1. Prueba con estados de deformación 1

Para este caso las cargas aplicadas son dos verticales y dos horizontales, una en cada nodo con incrementos iguales para lograr un estado uniforme de deformaciones. La configuración de cargas puede verse en la figura 7.5.

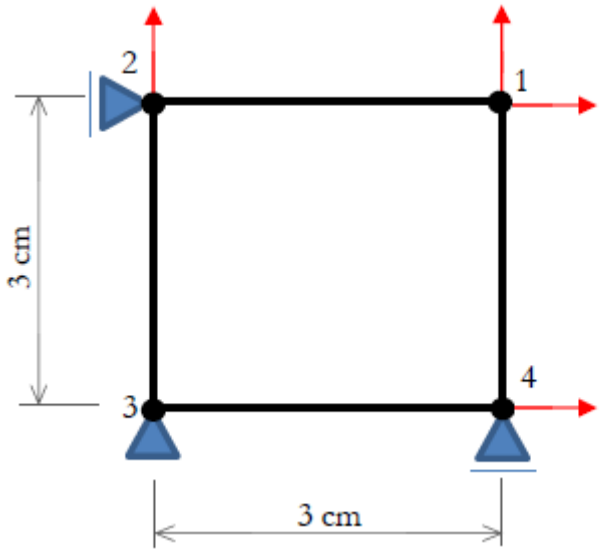

Figura 7.5: Configuración de carga 1. Rango de 500 a 9500 kgf, incrementos de 1000 kgf.

El estado de deformación final del elemento es similar al presentado en la figura 7.1.

En la figura 7.6 observamos la primera comprobación del funcionamiento del modelo constitutivo obtenido con la RNA. Los patrones de deformaciones usados como entrada son desconocidos para la red.

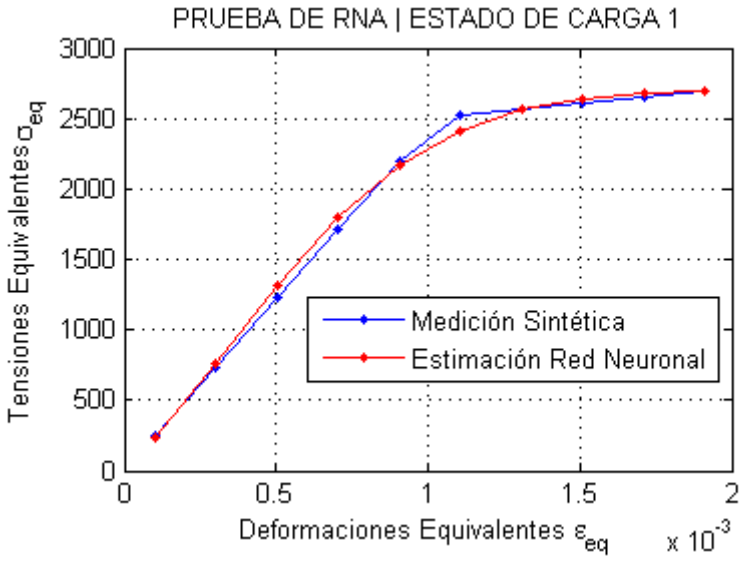

Figura 7.6: Estados para configuraciones de carga 1. Estado uniforme de deformaciones dentro del elemento.

7.5.2. Prueba con estados de deformación 2

Las cargas aplicadas son dos verticales y una horizontal con incrementos iguales. Por su asimetría los estados de deformación dentro del elemento no son uniformes (iguales), por lo tanto existen diferentes estados en cada punto de integración (son 4 puntos de Gauss, dispuestos de forma similar a los nodos, pero dentro del elemento). La configuración de cargas puede verse en la figura 7.7.

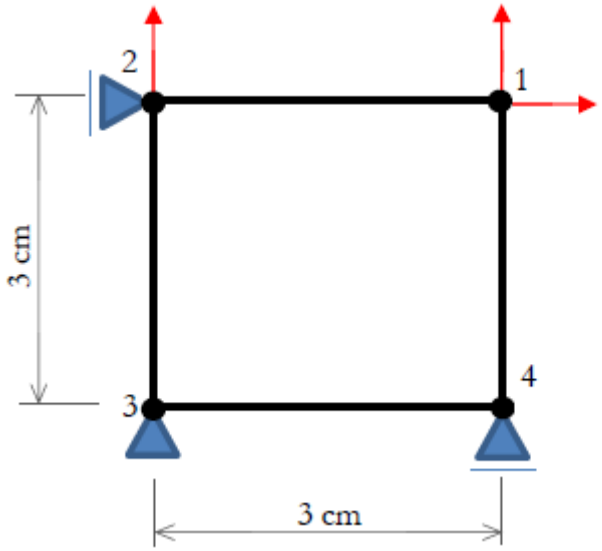

Figura 7.7: Configuración de carga 2. Rango de 500 a 9500 kgf, incrementos de 1000 kgf.

El estado de deformaciones del elemento se observa en la figura 7.8.

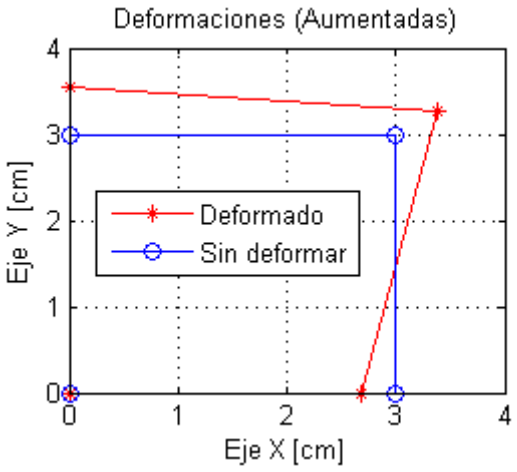

Figura 7.8: Gráfico de deformaciones del elemento para Caso de prueba 2.

En la figura 7.9 observamos la segunda comprobación del modelo de material elastoplástico. La curva azul identifica los estados sintéticos obtenidos por el que podría ser un probeta real, mientras que la curva roja los estados predichos por la RNA.

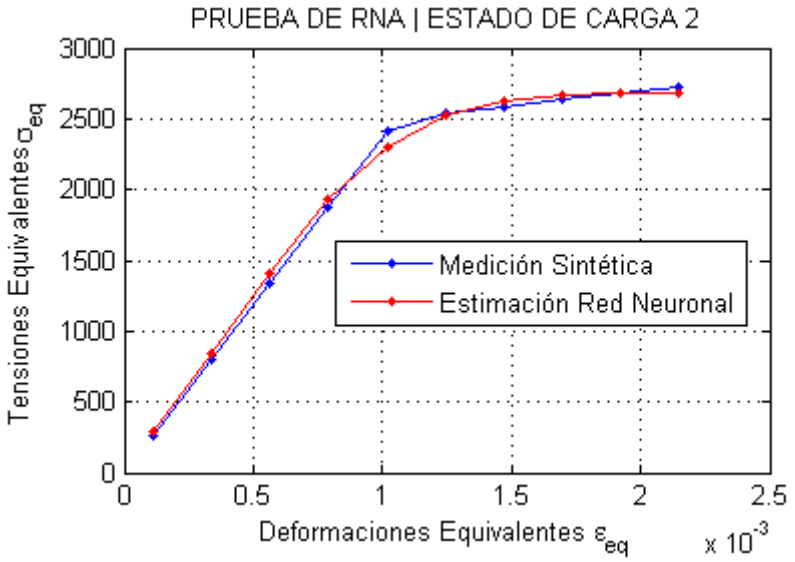

Figura 7.9: Estados para configuraciones de carga 2. Estado no-uniforme de deformaciones dentro del elemento. Datos para punto de Gauss 1 (abajo-izquierda).

7.5.3. Prueba con estados de deformación 3

La configuración de cargas para éste caso de prueba es la que vemos en la figura 7.10.

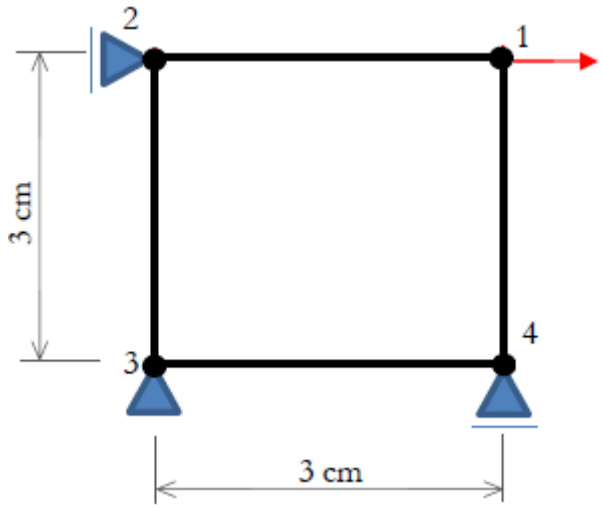

Figura 7.10: Configuración de carga 3. Rango de 0 a 20000 kgf, incrementos de 2000 kgf.

Las deformaciones del elemento plano sometido a las cargas de la figura 7.10 están ilustradas en la figura 7.11.

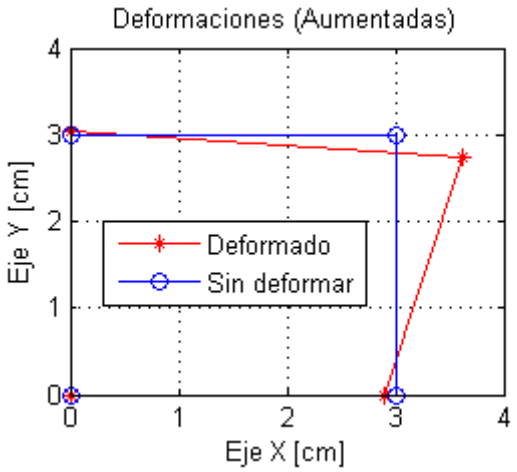

Figura 7.11: Gráfico de deformaciones del elemento para Caso de prueba 3.

Finalmente, en la figura 7.12 hallamos la tercera comprobación. Las cargas aplicadas son dos horizontales en los nodos 1 y 2, para lograr un efecto predominante de corte. Éste grupo de configuraciones de carga es probablemente la más alejada de aquellas con las cuales se entrenó la red.

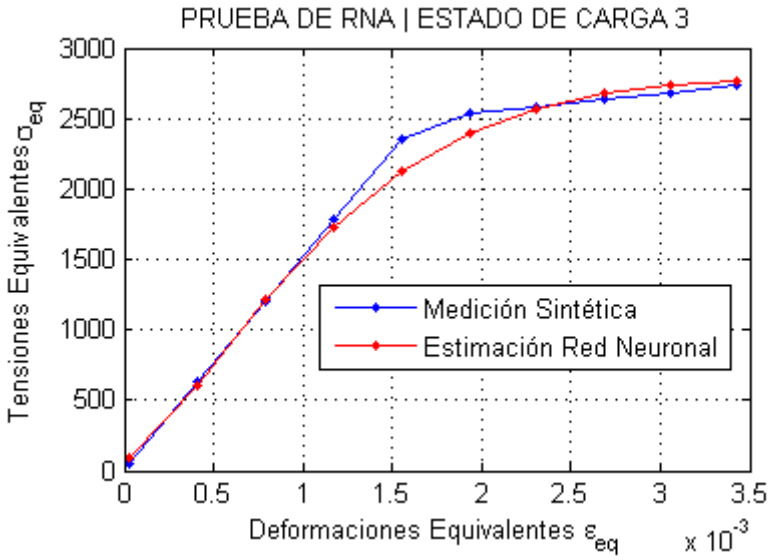

Figura 7.12: Estados para configuraciones de carga 3. Estado no-uniforme de deformaciones dentro del elemento. Datos para punto de Gauss 1 (abajo-izquierda).

7.6. Conclusiones

Las tres pruebas de la RNA sugieren que ante estados de deformación desconocidos para la red, es posible obtener estados de tensión aproximados que describen el material utilizado para extraer los datos de entrenamiento. Esto indica que a partir de datos de

deformaciones y tensiones de un material con comportamiento no-lineal también podemos obtener una estimación del modelo si empleamos redes de múltiples capas cuyos pesos puedan ajustarse al comportamiento no lineal del material. En las pruebas, la zona donde peores aproximaciones se obtienen es aquella donde se da el cambio de zona de elástico a plástico, donde la función de activación no puede aproximar fielmente el cambio discontínuo de una recta a otra.

Capítulo 8

Comentarios y Conclusiones

- Una buena razón para promover el uso de redes neuronales para aprender el comportamiento de un material es lograr una descripción acertada de algunos materiales cuyos modelos son complejos de determinar, como es el caso de los materiales compuestos, como el concreto reforzado con fibras, y vigas y columnas de concreto reforzado con acero.

- El Perceptrón utilizado para el caso de deformación plana posee 4 entradas y 4 salidas, las cuales fueron entrenadas simultáneamente. Una alternativa es entrenar una salida a la vez, lo que requeriría en un principio de 4 grupos de pesos, uno por salida. Esta forma de entrenar no fue implementada, pero permanece como idea para implementarse en futuros trabajos.

- En ensayos experimentales (triaxiales) de materiales en general es complejo obtener datos de deformación y tensión en cortante. La mayoría de estos datos están dados en términos de estados principales (sin datos de corte).

- Es posible entrenar una red neuronal simple para obtener, a partir de datos experimentales, un modelo de material matricial para materiales en su rango elástico. La técnica de enriquecimiento de datos permite disponer sólo de deformaciones y tensiones principales.

- Para aumentar la adaptabilidad del Perceptrón es posible programar un algoritmo que aumenta a demanda el número de neuronas en las capas ocultas hasta llegar a un número óptimo que evite el sobre-ajuste de la función aprendida por el perceptrón a los datos de las muestras, ya que éstas muchas veces contienen ruido que es preferible evitar en los resultados.

- La duración de los ensayos numéricos iterativos para entrenar el perceptrón y lograr un error aceptable entre la salida deseada y la salida actual de la red puede ser bastante prolongada en aquellos casos donde la cantidad de patrones a enseñar son numerosos. Una alternativa para optimizar los tiempos de aprendizaje es paralelizar las partes del código que permitan ser paralelizadas, como los productos escalares de las multiplicaciones de matriz por vector, donde repartiríamos el mismo vector a varios procesos y luego dividiríamos la las matrices y las repartiríamos por partes a cada proceso. Los resultados se retornarían después al proceso principal. De lograr un código escalable obtendríamos un ahorro en tiempo muy significativo.

- Una implementación para futuros trabajos es comprobar el comportamiento de la red para condiciones de borde diferentes a las estudiadas, donde se generarían otros estados de deformación que también deberían ser estudiados para validar el modelo de material con RNAs.
- El algoritmo original de back-propagation puede ser muy lento. Se han propuesto variaciones en este algoritmo para conseguir mayores velocidades de aprendizaje. Los investigadores de la Referencia [7] utilizaron el algoritmo RPROP [11] para entrenar sus modelos debido a su confiabilidad mejorada y a la relativamente baja cantidad de parámetros de control.

Capítulo 9

ANEXO 1: Código del Perceptrón con Backpropagation

El programa de entrenamiento y prueba es `Main.m`, que se halla dentro de la carpeta `tpEspV5`. Los patrones de prueba son obtenidos con la función `MainTP3v5iso.m`, y los caminos de carga son hallados modificando el código de la función `cargasUltimas.m`.

```
% MONOGRAFA FINAL DE CURSO
% ESTUDIANTE: FREDY MERCADO.
% REDES NEURONALES.
% PROFESOR: DR. SERGIO LEW.

clear all
close all
clc

%------------------------USUARIO-----------------------------%

TEST=0;  % MODO DE PRUEBA 1. MODO DE ENTRENAMIENTO 0.

%---PRUEBAS---%
if TEST==1
    load W
    ESTADOCARGA=3;
    if ESTADOCARGA==1
        load DATOSTEST
    elseif ESTADOCARGA==2
        load DATOSTEST2
    elseif ESTADOCARGA==3
        load DATOSTEST3
    end
end
%---PRE-PROCESO---%
NDATOSX=10;      % DATOS POR EJE COORDENADO.
INCANG=7;
ITERETA=1500;  % CADA ITERETA SE DISMINUYE ETA.
WPREV=1;  % 1. load W      0. W NUEVA.

%---PROCESO---%
NNEURONAS=[4;12;8;4];  % Cant. Neuronas en CAPA OCULTA.
ETACTUAL=0.0001;
BETA=1;           % CONSTANTE STEEPNESS PARAMETER. EN GRAL ES 1.
MAXITER=10000;    % NUMERO MAXIMO DE ITERACIONES.
```

```
37 MIN_ERMS=0.001;  % ERROR CUADRATICO MEDIO PERMISIBLE.
38
39 %--MOMENTUM---%
40 MOMENT=1;         % BACKPROPAGATION CON MOMENTUM TERM 1.SI 0.NO.
41 ALPHA=0.9;
42
43 %--POST-PROCESO---%
44 ITEPRINT=50;     % CANT. ITERACIONES IMPRESION DE PROGRESO.
45
46 %---------------------------USUARIO--------------------------%
47
48 if TEST==0
49     %-------------------------------------------------------------
50     %                        PRE-PROCESO
51     %    Genera patrones de entrenamiento y salida deseada.
52     %-------------------------------------------------------------
53     [EPSNN,SIGNN,EPSMAX,SIGMAX,EPS,SIG]=generaDatos3(INCANG);
54                                     % ENRIQUECE DATOS.
55
56     avisos(NNEURONAS,EPSNN,SIGNN);
57
58     %-------------------------------------------------------------
59     %                          MEMORIA
60     %-------------------------------------------------------------
61     DATOS=zeros(MAXITER,3);
62     NPATRONES=size(EPSNN,1);  % Cantidad de patrones.
63     NCW=length(NNEURONAS)-1;  % Nmero de Capas con pesos W.
64     MAXNN=max(NNEURONAS);  % Mximo Nmero de Neuronas en una capa.
65     MAXNN2=max(NNEURONAS(2:length(NNEURONAS)));
66     % Mximo Nmero de Neuronas en capas
67     % diferentes a la de entrada.
68     DELTAW=zeros(MAXNN,MAXNN,NCW);  % Mismas dimensiones que W.
69     SP=zeros(NPATRONES,NNEURONAS(NCW+1));
70     DELTA=zeros(MAXNN2,NCW);
71     EPSEQ=zeros(size(EPS,1),1);
72     SIGEQ=zeros(size(SIG,1),1);
73     SIGEQP=zeros(size(SIG,1),1);
74
75     %-------------------------------------------------------------
76     %                        PROCESO
77     %-------------------------------------------------------------
78
79     if WPREV==0
80         [W]=generaW(MAXNN,NCW,NNEURONAS);
81                                   % Genera W de pesos aleatorios.
82     elseif WPREV==1
83         load W
84     end
85
86     ETA=ETACTUAL;
87     for ITE=1:MAXITER
88         if rem(ITE,ITERETA)==0
89             ETA=ETACTUAL*0.75;
90             ETACTUAL=ETA;
91         end
92         SC=zeros(MAXNN2,NCW);
93         for P=1:NPATRONES
94             EPATRON=EPSNN(P,:);
```

```
95
96              %-------------------------------------------------------
97              %                 CALCULO DE SALIDAS
98              %-------------------------------------------------------
99
100             for C=1:NCW % Capa | N Capas que contienen pesos W
101                 NEURIN=NNEURONAS(C); % Neuronas de entrada a Capa C
102                 NEUROUT=NNEURONAS(C+1); % N de salida de Capa C
103                 WCAPA=W(1:NEUROUT,1:NEURIN,C); % Matriz pesos Capa C
104                 if C==1
105                     EC=EPATRON'; % Para Capa 1 entrada=patrn.
106                 else
107                     EC=SC(1:NEURIN,C-1); % Para otras capas
108                     % entrada es la salida
109                     % de la capa anterior.
110                 end
111                 HC=WCAPA*EC; % HCAPA SUMATORIA NEURONA.
112                 SC(1:NEUROUT,C)=tanh(BETA*HC); % SALIDAS CAPAS.
113             end
114
115             SP(P,:)=SC(1:NEUROUT,NCW)';
116
117             %-------------------------------------------------------
118             %                   BACKPROPAGATION
119             %-------------------------------------------------------
120
121             for C=NCW:-1:1
122                 NEURIN=NNEURONAS(C); % Neuronas de entrada a Capa C
123                 NEUROUT=NNEURONAS(C+1); % N de salida de Capa C
124                 if C==NCW
125                     WC1=0;
126                     DELC1=0;
127                 else
128                     WC1=W(1:NNEURONAS(C+2),1:NNEURONAS(C+1),C+1);
129                     DELC1=DELTA(1:NNEURONAS(C+2),C+1);
130                 end
131                 [DELTA(1:NEUROUT,C)]=calcDelta(NNEURONAS,C,NCW,...
132                     BETA,SC(1:NEUROUT,C),SIGNN(P,:),WC1,DELC1);
133                 [DELTAW(1:NEUROUT,1:NEURIN,C)]=calcDeltaW(C,...
134                     EPATRON,SC,ETA,DELTA(1:NEUROUT,C),NEURIN,...
135                     MOMENT,ALPHA,DELTAW(1:NEUROUT,1:NEURIN,C));
136             end
137             W=W+DELTAW;
138         end
139
140         [ERMS]=errorCM3(SP,SIGNN);
141
142         if rem(ITE,ITEPRINT)==0
143             fprintf('ITE=%d, ERMS=%d,\n SP=%f, SIGNN=%f\n',ITE,...
144                 ERMS,SP(1,1),SIGNN(1,1))
145             SIGP=SP*SIGMAX;
146
147             for i=1:size(EPS,1)
148                 [EPSEQ(i),SIGEQ(i)]=equivalent(EPS(i,:),SIG(i,:));
149                 [AAA,SIGEQP(i)]=equivalent(EPS(i,:),SIGP(i,:));
150             end
151
152             hold on
```

```
153             if ITE>50
154                 delete(GRAF)
155             end
156             GRAF=plot(EPSEQ,SIGEQ,'b.',EPSEQ,SIGEQP,'r.');
157             grid on
158             pause(1)
159         end
160
161         [SUMERROR]=SumError2(SP,SIGNN);
162         DATOS(ITE,1)=ITE;
163         DATOS(ITE,2)=ERMS;
164         DATOS(ITE,3)=0;
165
166         if ERMS<MIN_ERMS
167             fprintf('ITE=%d, ERMS=%d\n',ITE,ERMS)
168             break;
169         end
170
171     end % PROCESO.
172
173     figure
174     plot(EPSEQ,SIGEQ,'b.',EPSEQ,SIGEQP,'r.')
175     title('ESTADOS EQUIVALENTES')
176     xlabel('Deformaciones Equivalentes \epsilon_{eq}')
177     ylabel('Tensiones Equivalentes \sigma_{eq}')
178     legend('Medicin Sinttica','Estimacin Red Neuronal')
179     grid on
180
181     figure
182     plot(EPSEQ,SIGEQ,'b.')
183     title('ESTADOS SINTTICOS PARA ENTRENAMIENTO DE RNA')
184     xlabel('Deformaciones Equivalentes \epsilon_{eq}')
185     ylabel('Tensiones Equivalentes \sigma_{eq}')
186     grid on
187
188     figure
189     plot(EPSEQ,SIGEQP,'r.')
190     title('ESTADOS ALCANZADOS CON RNA TRAS ENTRENAMIENTO')
191     xlabel('Deformaciones Equivalentes \epsilon_{eq}')
192     ylabel('Tensiones Equivalentes \sigma_{eq}')
193     grid on
194
195 elseif TEST==1
196     %-------------------------------------------------------------
197     %                          PRUEBAS
198     %
199     % TOMAMOS DATOS GENERADOS POR MainTP3v5iso.m, EL CUAL ES UN
200     % PROGRAMA PARA OBTENER EPSPNN, SIGPNN, EPSEQPNN, SIGEQPNN,
201     % MATRICES DE DEFORMACIONES Y TENSIONES, ESTADOS Y
202     % VALORES EQUIVALENTES.
203     %-------------------------------------------------------------
204
205     [EPSNN,SIGNN,EPSMAX,SIGMAX,EPS,SIG]=generaDatos3(INCANG);
206     [SIGEQTP]=testNN(W,EPST,SIGT,NNEURONAS,BETA,EPSMAX,SIGMAX);
207
208     figure
209     plot(EPSEQT,SIGEQT,'b.-',EPSEQT,SIGEQTP,'r.-')
210     title(['PRUEBA DE RNA | ESTADO DE CARGA ' num2str(ESTADOCARGA)])
```

```
211     xlabel('Deformaciones Equivalentes \epsilon_{eq}')
212     ylabel('Tensiones Equivalentes \sigma_{eq}')
213     legend('Medicin Sinttica','Estimacin Red Neuronal')
214     grid on
215 end
```

Capítulo 10

ANEXO 2: Algoritmo autoprogresivo

En las primeras aplicaciones de redes neuronales al modelado de relaciones constitutivas éstas fueron entrenadas con datos tomados directamente de pruebas experimentales. La modelación constitutiva a partir de ensayos es un problema inverso. En este caso la entreda (tensiones aplicadas) y la salida (deformaciones medidas) son conocidas y el sistema, en este caso la relación constitutiva, debe ser determinada. Con los métodos matemáticos convencionales esto se logra desarrollando un modelo matemático que reproduzca lo más cercanamente posible a las mediciones experimentales.

La ventaja de extraer la información directamente del comportamiento del material es que no hay necesidad de idealizar. Es más, como los datos provienen de datos del material mismo, es seguro asumir que cumple con las leyes de la mecánica. También existen desventajas. Los ensayos de materiales están diseñados para representan un punto del material. Como consecuencia, el estado de tensiones y deformaciones dentro de la probeta debe ser lo más uniforme posible. La muestra es sometida a una configuración de tensiones y se asume que todos los puntos dentro de la muestra poseen la misma configuración (el mismo estado), por lo tanto, los datos generados a partir de estas pruebas posee información sólo a lo largo de la zona con dicha configuración. Los datos de un sólo ensayo no es suficiente para entrenar un modelo de material implementado con redes neuronales. Se necesita información del comportamiento del material sobre toda la región de interés en el espacio de tensiones para poder entrenar un modelo constitutivo basado en redes neuronales con capacidades de generalización. Esto requiere una serie de ensayos especialmente diseñados con configuraciones de tensiones que cubran razonablemente toda la región de interés en el espacio de tensiones, lo cual no es práctico en la mayoría de los casos.

Los ensayos de materiales no son la única fuente de información sobre el comportamiento de un material. Existen otras fuentes potenciales, como son las pruebas estructurales, donde se miden los desplazamientos de una estructura sometida a fuerzas conocidas. Este problema inverso es más complejo que los ensayos de materiales. Las fuerzas y desplazamientos son las entradas y salidas conocidas del sistema y se deben hallar las propiedades constitutivas del material que conforma la estructura. A diferencia de las pruebas a los materiales, las cuales inducen (idealmente) un estado de tensión uniforme dentro de la muestra, las pruebas estructurales inducen un estado de tensión

y deformación no uniforme. Ya que los puntos dentro de la estructura siguen estados de tensión diferentes se tiene mucha más información de una sola prueba estructural que de un ensayo del material, donde todos los puntos tienen el mismo estado.

Extraer la información de las propiedades materiales de las pruebas estructurales es un problema extremadamente difícil con los métodos matemáticos convencionales. Probablemente por ésto no se ha intentado en el pasado. Por otro lado, los métodos de computación como las RNA (Redes Neuronales Artificales) son ideales para este tipo de problemas inversos. El algoritmo autoprogresivo es un método para entrenar una red neuronal para que aprenda propiedades constitutivas de los materiales a partir de pruebas estructurales (ver Ghaboussi, Pecknold, Zhang y HajAli, 1998) [5].

El algoritmo autoprogresivo es útil para ser empleado en forma híbrida para modelación de estructuras mediante Elementos Finitos. Es una de las posibilidades actuales para fusionar redes neuronales con dicho método aplicado sobre todo a la solución de problemas con materiales con comportamientos no lineales [6].

Capítulo 11

ANEXO 3: Uso de RNAs para modelar materiales en Elementos Finitos

El objetivo de construir modelos constitutivos para materiales de ingeniería es usar el modelo con un procedimiento de solución, como es el método de los Elementos Finitos, de tal forma que el comportamiento estructural pueda ser predicho de forma confiable. En general, en el método de Elementos Finitos el modelo de material sirve para dos propósitos:

1. Evaluar la matriz constitutiva C y con ella calcular las matrices de rigidez elementales K como
$$[K] = \int_V [B]^T [C][B] \, dV \tag{11.1}$$
donde B es la matriz de deformaciones-desplazamientos y V el volumen del elemento.

2. Determinar el estado actual de tensiones que corresponden con un estado actual de deformaciones.
$$\{\sigma\} = [C]\{\varepsilon\} \tag{11.2}$$
donde $\{\sigma\}$ y $\{\varepsilon\}$ son los vectores de tensiones y deformaciones, respectivamente.

Para el modelo representado por la Eq. 11.2, la incorporación de la RNA puede realizarse fácilmente al tener deformaciones como entradas y tensiones como salidas.

11.1. Implementación de modelos de material basados en RNAs

11.1.1. Uso directo de RNAs

El modelo constitutivo en un programa de elementos finitos se utiliza en tres etapas: la formación de las matrices de rigidez elementales, el cálculo de las fuerzas equivalentes a tensiones y el cálculo de las tensiones elementales. Hay tres tipos de análisis que se pueden resolver:

1. Análisis estático.
2. Análisis dinámico con integración implícita en el tiempo.
3. Análisis dinámico con integración explícita en el tiempo.

Aparentemente, si un esquema Newton-Raphson es utilizado se requiere de la forma explícita de la matriz C para resolver los dos primeros problemas, excepto el tercero. Sin embargo, para análisis estáticos es bien sabido que la formulación explícita de la matriz de rigidez no es necesaria si se utilizan esquemas iterativos como el Gradiente Conjugado como métodos de solución. Se puede entonces utilizar un modelo de material basado en RNAs para análisis estáticos y dinámicos con integración explícita en el tiempo, pero no se puede para problemas dinámicos con integración implícita [15].

11.1.2. Uso indirecto de RNAs

En la sección anterior se concluyó que el empleo directo de RNAs requiere de la implementación de métodos especiales. No obstante, la flexibilidad de las RNAs permite evaluar la matriz de tensiones-deformaciones C a través del entrenamiento con un grupo de datos aumentado con información adicional.

La primera forma de entrenamiento se basa en que la información capturada en la red neuronal está determinada exclusivamente por la información contenida en los datos de entrenamiento. Si la información de la matriz de tensiones-deformaciones está apropiadamente representada el modelo entrenado con estos datos debería contener la información relevante.

La segunda forma de entrenamiento consiste en que los elementos de la matriz tensiones-deformaciones necesitan ser incluidos explícitamente en el esquema de salida de la red. Ver Figura 11.1. [15].

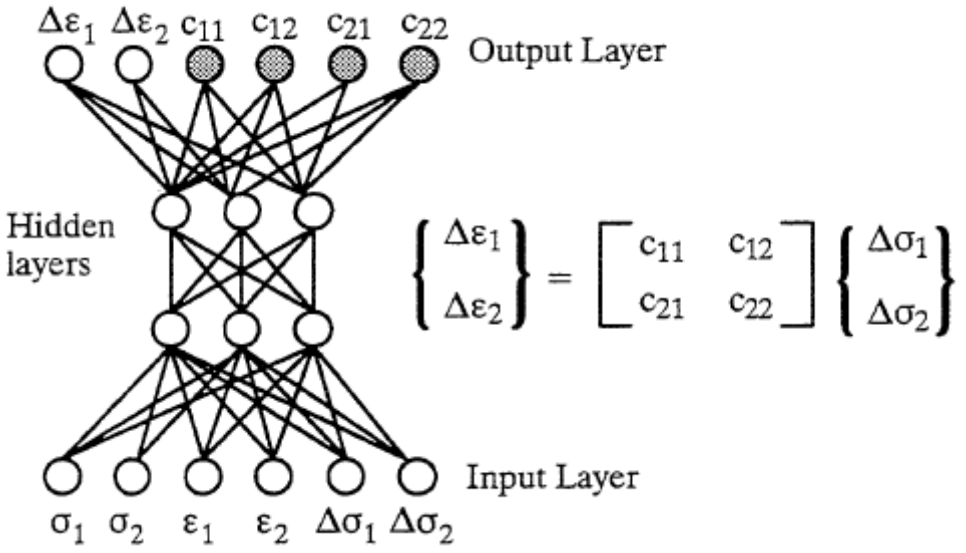

Figura 11.1: Topología de Red Neuronal para determinar matriz de tensión-deformación.

11.2. Entrenamiento de la red

El entrenamiento de redes neuronales para modelar relaciones constitutivas se puede realizar mediante una aproximación directa, en donde la red es entrenada a partir de mediciones de tensiones y deformaciones de un material (de concreto plano, por ejemplo, a lo largo de diferentes caminos de carga biaxial). Como sustento de lo anterior se pueden consultar las citas 3-5, 8 y 10 de la referencia [6]. Una de las desventajas de ésta forma de entrenar la red es la enorme cantidad de datos experimentales que podrían necesitarse para el entrenamiento cuando el comportamiento del material es bastante complejo. Adicionalmente, podrían haber serias dificultades en algunos casos para mantener condiciones espaciales homogeneas a nivel macroscopico en un especimen de prueba, en particular cuando éste se encuentre sometido a caminos multiaxiales de tensión y deformación con ciclos de carga y descarga. En verdaderas pruebas triaxiales, pruebas de corte y de torsión, se deben tomar precauciones especiales para segurar la uniformidad del campo de tensiones [6].

Bibliografía

[1] Chen, T., and Chen, H. Universal approximation to non-linear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Trans. Neural Networks*, 6 (4) (1995), 911–917.

[2] Drakos, S. I., and Pande, G. N. A neural network equivalent of hardening soil model of plaxis. *Numerical Methods in Geotechnical Engineering* (2006), 651–656.

[3] Flood, I., and Kartam, N. Neural networks in civil engineering. i: principles and understanding. *Journal of Computing in Civil Engineering*, 8(2) (1994), 131–147.

[4] Freitag, S., Graf, W., and Kaliske, M. A material description based on recurrent neural networks for fuzzy data and its application within the finite element method. *Computers and Structures*, 124 (2013), 29–37.

[5] Ghaboussi, J. *Chapter 4: Advances in Neural Networks in Computational Mechanics and Engineering (book).* University of Illinois at Urbana Champaign.

[6] Ghaboussi, J., Pecknold, D., Zhang, M., and Haj-Ali, R. Autoprogressive training of neural network constitutive models. *Int. J. Numer. Meth. Engng*, 42 (1998), 105–126.

[7] Hashash, Y. M. A., Jung, S., and Ghaboussi, J. Numerical implementation of a neural network based material model in finite element analysis. *Int. J. Numer. Meth. Engng*, 59 (2004), 989–1005.

[8] Hertz, J., Krogh, A., and Palmer, R. G. *Introduction to the theory of neural computation*, vol. 1. Addison-Wesley, 1990.

[9] Lefik, M., Boso, D., and Schrefler, B. Artificial neural networks in numerical modelling of composites. *Computational Methods in Applied Mechanics Engineering*, 198 (2009), 1785–1804.

[10] Lin, Y., and Chen, X.-M. A critical review of experimental results and constitutive descriptions for metals and alloys in hot working. *Materials and design*, 32 (2011), 1733–1759.

[11] Riedmiller, M., and Braun, H. A. A direct adaptive method for faster backpropagation learning: The rprop algorithm. *IEEE International Conference in Neural Networks, IEEE: San Francisco, New York* (1993), 586–591.

[12] Shin, H. S., and Pande. Enhancement of data for training neural network based constitutive models for geomaterials. *Proc. of the Eighth Intl. Symp. on Numerical Models in Geomechanics (NUMOG VIII), Rome, Italy, April* (2002), 141–146.

[13] Sun, Y., Zeng, W., Zhao, Y., Qi, Y., Ma, X., and Han, Y. Development of constitutive relationship model of ti600 alloy using artificial neural network. *Computational Materials Science*, 48 (2010), 686–691.

[14] Swingler, K. *Applying Neural Networks: A Practical Guide*. Morgan Kaufmann Publishers, California, 1996.

[15] Wu, X., and Ghaboussi, J. *Neural Network-based Material Modelling*. University of Illinois, 1995.