Programming ‘Language’ {

Este código implementa una red neuronal artificial con una arquitectura de 4 entradas, 10 neuronas ocultas y 3 salidas. El objetivo principal es predecir las salidas para las observaciones de la flor Iris, utilizando un conjunto de datos de entrenamiento con 30 observaciones. La red utiliza una función sigmoide tanto en la capa oculta como en la capa de salida.

}

PRIMERA PARTE‘Parámetros de la red’

```
n_entradas = 4;  // Número de entradas
n_ocultas = 10;  // Neuronas en la capa oculta
n_salidas = 3;   // Neuronas en la capa de salida
n_num_dat_ent = 30 //numero de datos de entrenamiento
```

* **n_entradas = 4:** Número de características de entrada (las medidas de las flores Iris).
* **n_ocultas = 10:** Número de neuronas en la capa oculta.
* **n_salidas = 3:** Número de neuronas en la capa de salida, correspondientes a las tres clases (setosa, versicolor, virginica).
* **n_num_dat_ent = 30:** Número de datos de entrenamiento (30 muestras de Iris.

SEGUNDA PARTE‘Funciones de activación’

```
//% Función de activación sigmoide
function y = sigmoid(x)
    y = 1 ./ (1 + exp(-x));
endfunction

//% Derivada de la sigmoide
function y = sigmoid_derivada(x)
    y = sigmoid(x) .* (1 - sigmoid(x));
endfunction
```

- **sigmoid(x):** Función sigmoide utilizada como función de activación.
- **sigmoid_derivada(x):** Derivada de la función sigmoide, utilizada durante el proceso de retropropagación para actualizar los pesos.

}

TERCERA PARTE‘Inicialización de Pesos y Sesgos’ {

```
W1 = rand(n_entradas, n_ocultas);  // Pesos capa oculta
b1 = rand(1, n_ocultas);           // Sesgos capa oculta
W2 = rand(n_ocultas, n_salidas);   // Pesos capa salida
b2 = rand(1, n_salidas);           // Sesgos capa salida
```

< Se inicializan aleatoriamente los pesos (W1 y W2) y los sesgos (b1 y b2) para las capas ocultas y de salida. >

}

Martha Elena Inda Ramos

```
//30 datos de las observaciones IRIS y sus etiquetas
IRIS_DATA=[
[5.1,3.5,1.4,0.2];[4.9,3,1.4,0.2];[4.7,3.2,1.3,0.2];[4.6,3.1,1.5,0.2];[5,3.6,1.4,0.2];
[5.4,3.9,1.7,0.4];[4.6,3.4,1.4,0.3];[5,3.4,1.5,0.2];[4.4,2.9,1.4,0.2];[4.9,3.1,1.5,0.1];
[6.4,3.2,4.5,1.5];[6.9,3.1,4.9,1.5];[5.5,2.3,4,1.3];[6.5,2.8,4.6,1.5];[5.7,2.8,4.5,1.3];
[6.3,3.3,4.7,1.6];[4.9,2.4,3.3,1];[6.6,2.9,4.6,1.3];[5.2,2.7,3.9,1.4];[5,2,3.5,1];
[6,3,4.8,1.8];[6.9,3.1,5.4,2.1];[6.7,3.1,5.6,2.4];[6.9,3.1,5.1,2.3];[5.8,2.7,5.1,1.9];
[6.8,3.2,5.9,2.3];[6.7,3.3,5.7,2.5];[6.7,3,5.2,2.3];[6.3,2.5,5,1.9];[6.5,3,5.2,2]
];
Y_DATA=[
[1,0,0];[1,0,0];[1,0,0];[1,0,0];[1,0,0];[1,0,0];[1,0,0];[1,0,0];[1,0,0];[1,0,0];
[0,1,0];[0,1,0];[0,1,0];[0,1,0];[0,1,0];[0,1,0];[0,1,0];[0,1,0];[0,1,0];[0,1,0];
[0,0,1];[0,0,1];[0,0,1];[0,0,1];[0,0,1];[0,0,1];[0,0,1];[0,0,1];[0,0,1];[0,0,1]
];
// Datos de entrenamiento (ejemplo)
//X = rand(n_num_dat_ent, n_entradas);  // 100 muestras, 4 características
//Y = rand(n_num_dat_ent, n_salidas);   //% 100 etiquetas, 3 salidas

X=IRIS_DATA;
Y=Y_DATA;
//mostrar entradas
disp("Datos de entrada")
disp(cat(2,X,Y))
}
```

CUARTA PARTE‘Datos de entrenamiento’

CUARTA PARTE‘Datos de entrenamiento’ {

< **IRIS_DATA:** Conjunto de datos de las características de las flores Iris (seis características en total: largo y ancho del sépalo y pétalo). >

}

< **Y_DATA:** Las etiquetas en formato de codificación one-hot, donde cada fila indica a que clase pertenece la observación (1 para Setosa, 2 para Versicolor, 3 para Virginica). >

}

Martha Elena Inda Ramos

1

```
//entrenamiento
disp("Entrenamiento:")
//% Hiperparámetros
tasa_aprendizaje = 0.1;
max_iter = 1000;

//% Entrenamiento
for iter = 1:max_iter
    //% Propagación hacia adelante
    b1_expanded = repmat(b1, n_num_dat_ent,1);
    //Expande b1[1,10] para que sea [30,10]
    //para sumar a esto: X * W1 = [30,4]*[4,10]=[30,10]
    //suma de las entradas ponederadas + los sesgos
    Z1 = X * W1 + b1_expanded
    //A1 salidas de la capa oculta
    A1 = sigmoid(Z1);
    //A2 salidas de la capa de salida
    b2_expanded = repmat(b2, n_num_dat_ent,1);
    Z2 = A1 * W2 + b2_expanded;
    A2 = sigmoid(Z2);
```

```
    //Cálculo del error
    error = Y - A2;

    //Retropropagación
    dZ2 = error .* sigmoid_derivada(Z2);
    dW2 = A1' * dZ2;
    db2 = sum(dZ2, 1);

    dZ1 = (dZ2 * W2') .* sigmoid_derivada(Z1);
    dW1 = X' * dZ1;
    db1 = sum(dZ1, 1);

    //% Actualizar pesos y sesgos
    W2 = W2 + tasa_aprendizaje * dW2;
    b2 = b2 + tasa_aprendizaje * db2;
    W1 = W1 + tasa_aprendizaje * dW1;
    b1 = b1 + tasa_aprendizaje * db1;
end
```

2

}

QUINTA PARTE‘Entrenamiento de la Red’

Programming Language

QUINTA PARTE‘Entrenamiento de la Red’

Hiperparámetros: tasa_aprendizaje = 0.1 (indica cuán rápido ajustamos los pesos) y max_iter = 1000 (máximo número de iteraciones para el entrenamiento).

Propagación hacia adelante:

- Se calculan las salidas de las capas, primero en la capa oculta (A1) y luego en la capa de salida (A2), utilizando la función sigmoide.

Cálculo del error: Se calcula el error entre la salida esperada (Y) y la salida predicha (A2).

Retropropagación:

- Se calcula la derivada del error en cada capa utilizando la regla de la cadena.

- Se actualizan los pesos (W1, W2) y los sesgos (b1, b2) con el gradiente calculado.

Martha Elena Inda Ramos

SEXTA PARTE‘Probar la Red’ {

```
b1_exp=repmat(b1,n_num_dat_ent,1)
b2_exp=repmat(b2,n_num_dat_ent,1)
Y_pred = sigmoid(sigmoid(X * W1 + b1_exp) * W2 + b2_exp);
```

< Después del entrenamiento, la red se prueba con los mismos datos de entrada, utilizando los pesos y sesgos ajustados. Se realiza la propagación hacia adelante nuevamente y se muestran las predicciones de salida. >

1

Expansión de sesgos (b1_exp y b2_exp): Se utilizan las funciones repmat para expandir los sesgos (b1 y b2) de las capas de la red. La idea es replicar los sesgos para que coincidan con el número de muestras de entrada (30), de modo que puedan sumarse correctamente a las activaciones.

2

Propagación hacia adelante: La propagación hacia adelante se realiza calculando las activaciones de la red con los pesos y sesgos entrenados. Primero se multiplica el conjunto de entradas X por los pesos de la capa oculta W1, se le suma el sesgo b1_exp, y luego se pasa a través de la función sigmoide para obtener las activaciones de la capa oculta (A1).

Continuación

Después, las activaciones de la capa oculta A1 se multiplican por los pesos de la capa de salida W2, se le suma el sesgo b2_exp, y se pasa nuevamente por la función sigmoide para obtener las predicciones de la salida Y_pred.

SÉPTIMA PARTE‘Salida de predicciones’

{

```
disp("Predicciones:");
disp(cat(2,X,fix(Y_pred+0.5)));
//disp(Y_pred);
```

Las predicciones de la red. Para convertir la salida continua de la sigmoide en valores discretos de 0 o 1, se redondean las predicciones con fix(Y_pred + 0.5), lo que ayuda a que las salidas sean valores cercanos a 0 o 1, representando las clases de forma codificada como one-hot (por ejemplo, [0, 0, 1] para la clase 3, [0, 1, 0] para la clase 2, y [1, 0, 0] para la clase 1.

}