



REPORTE UNIDAD 3

VIDEO NO.9

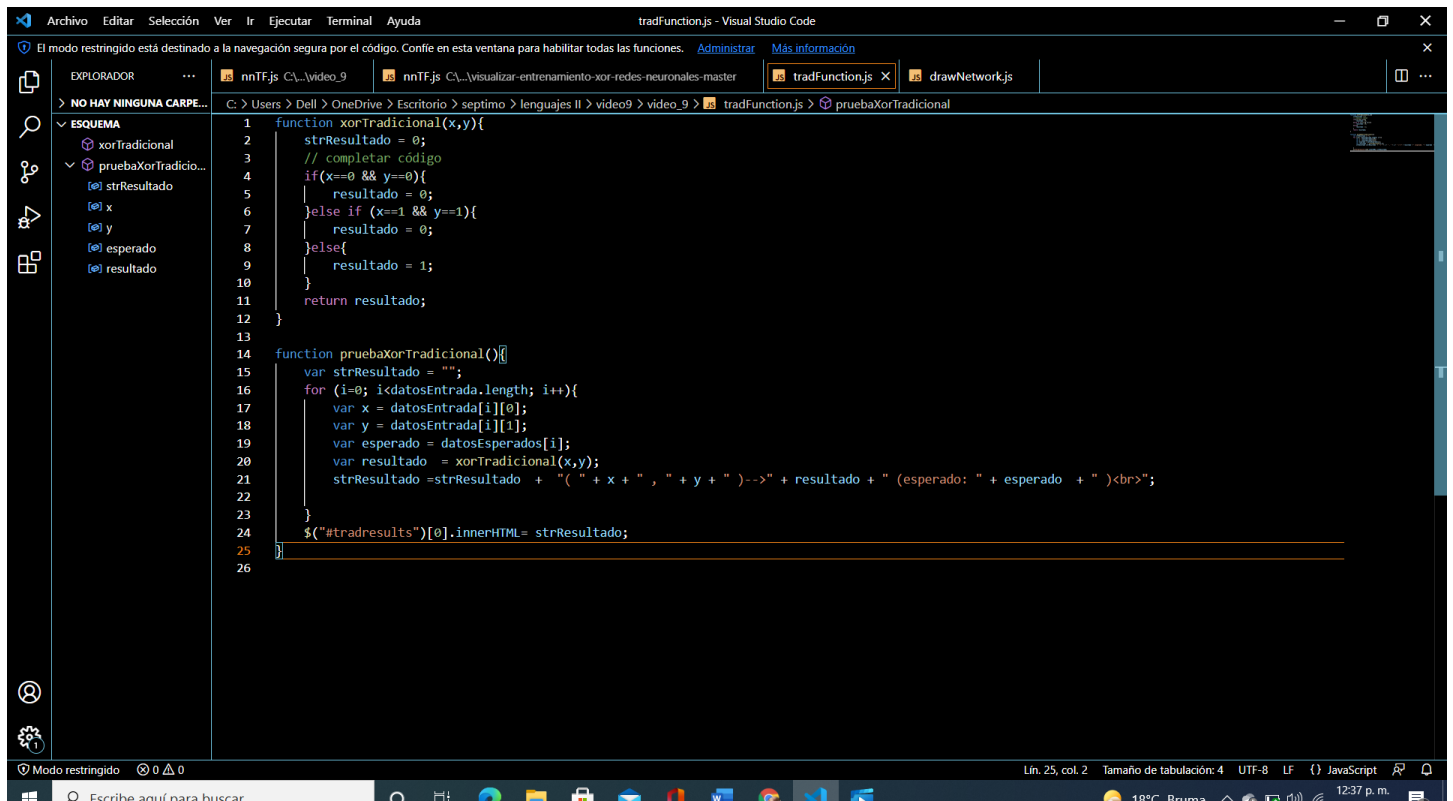
Lenguajes y Autómatas II



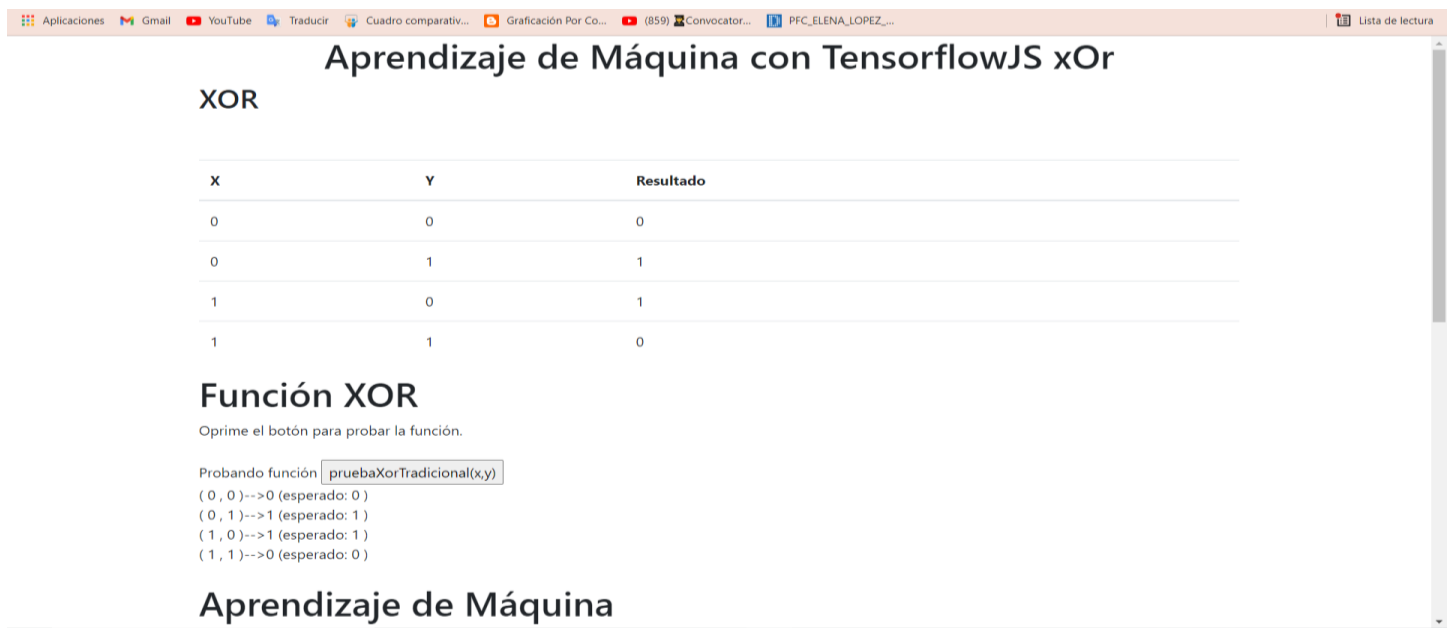
09 DE NOVIEMBRE DE 2021
PONCE MARTÍNEZ YOLLOTL AMEYALLI
AQUINO SEGURA ROLDAN

En esta practica tenemos un ejemplo en el cual veremos cómo está programado a través de una red neuronal una compuerta lógica como el XOR. El XOR nos debe de regresar verdadero o 1 en aquellos casos donde X o Y son verdadero, pero no en aquellos donde los dos son falsos o donde los dos son verdaderos.

En la siguiente imagen podremos ver como se vería ejecutando un código tradicional el cuál esta en el archivo tradFuctions.js



Que sería esta parte de nuestro index.html



El código para el aprendizaje de maquina sería el siguiente (Casi todo esta comentado en el código también). Esto se encontrar en el archivo nnTF.js

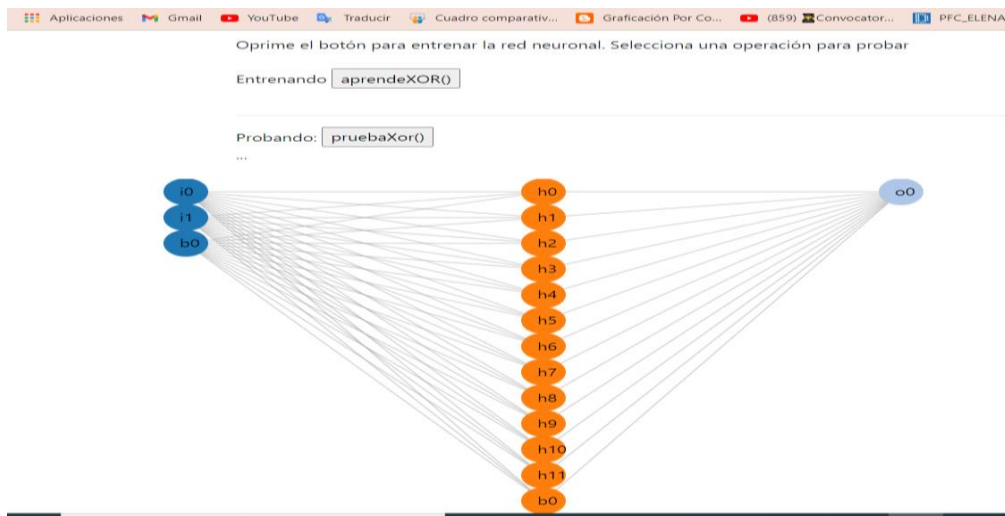
Tiene cierto numero de neuronas en la capa oculta (en este caso 12), tiene una tasa de aprendizaje que quiere decir que va a ir evolucionando con cierta velocidad hacia el punto optimo y que va a tener como datos de entrada del entrenamiento los valores que especificaremos en el problema.

```
nnTF.js - Visual Studio Code
El modo restringido está destinado a la navegación segura por el código. Confíe en esta ventana para habilitar todas las funciones. Administrar Más información
EXPLORADOR
nnTF.js
> NO HAY NINGUNA CARPE...
ESQUEMA
NUM_NEURONAS_...
TASA_DE_APRENDI...
noHeParadoAntes
ciclosDeAprendizaje
datosEntrada
1 //Datos de Configuración
2 //TODO: Juega con el número de neuronas de la capa oculta y la tasa de aprendizaje, observa que sucede
3 const NUM_NEURONAS_CAPA_OCULTA = 12;
4 const TASA_DE_APRENDIZAJE = 0.1;
5 var noHeParadoAntes = true;
6 var ciclosDeAprendizaje = 0;
7 // datos para entrenar
```

Tenemos las etiquetas es decir, el resultado que estoy esperando, definiremos los datos de entrada que tenemos que transformar en tensores que son "Arreglos de datos", tenemos tensores de los pesos, del bias.

```
nnTF.js - Visual Studio Code
El modo restringido está destinado a la navegación segura por el código. Confíe en esta ventana para habilitar todas las funciones. Administrar Más información
EXPLORADOR
nnTF.js
> NO HAY NINGUNA CARPE...
ESQUEMA
NUM_NEURONAS_...
TASA_DE_APRENDI...
noHeParadoAntes
ciclosDeAprendizaje
datosEntrada
datosEsperados
tensorEntrada
tensorEsperado
pesosCapaUno
biasCapaUno
pesosCapaDos
biasCapaDos
model
hiddenLayer
6 var ciclosDeAprendizaje = 0;
7 //! datos para entrenar
8 //!datos de entrada (XOR)
9 const datosEntrada = [[0,0],[0,1],[1,0],[1,1]];
10
11 //!resultados esperrados para cada dato de entrada
12 const datosEsperados = [[0],[1],[1],[0]];
13
14 //inicializacion de tensores, peso y bias(utilizamos los datos y los convertimos)
15 const tensorEntrada = tf.tensor2s(datosEntrada, [4,2]);
16 const tensorEsperado = tf.tensor2d(datosEsperados, [4,1]);
17
18 //las variables se registran en tensorflow como variables entrenable
19 const pesosCapaUno = tf.variable(inicializaPesos([2, NUM_NEURONAS_CAPA_OCULTA], 2));
20 const biasCapaUno = tf.variable(tf.scalar(0));
21 const pesosCapaDos = tf.variable(inicializaPesos([NUM_NEURONAS_CAPA_OCULTA,1],NUM_NEURONAS_CAPA_OCULTA));
22 const biasCapaDos = tf.variable(tf.scalar(0));
```

Básicamente consistirían en cuanto pesa cada una de estas conexiones.



Después de definir el modelo que en este caso estamos usando el Core API, lo entrenamos a través de un optimizador gradiente de descenso. Se tendrán algunas funciones las cuales nos van a ayudar a inicializar los pesos de la red neuronal y el cálculo del costo sobre el cual voy a hacer la optimización. Este último es el resultado que empieza a dar la red vs el resultado que se está esperando y se debe de optimizar de tal manera que se vaya reduciendo este costo.

Visual Studio Code - nnTF.js - Visual Studio Ayuda

El modo restringido está destinado a la navegación segura por el código. Confíe en esta ventana para habilitar todas las funciones. [Administrar](#) [Más información](#)

EXPLORADOR

NO HAY NINGUNA CARPE... C: > Users > Dell > OneDrive > Escritorio > septimo > lenguajes II > video9 > nnTF.js > ...

ESQUEMA

- NUM_NEURONAS...
- TASA_DE_APRENDI...
- noHeParadoAntes
- ciclosDeAprendizaje
- datosEntrada
- datosEsperados
- tensorEntrada
- tensorEsperado
- pesosCapaUno
- biasCapaUno
- pesosCapaDos
- biasCapaDos
- model
 - hiddenLayer
 - tf.tidy() callback
 - optimizador
 - inicializacion
 - calculaCosto
- entrena
 - regresaCosto
 - costo
 - i

```
23   
24 //definicion del modelo de red neuronal con TensorFlow.js core API  
25 //son funciones que toman uno o mas tensores y devuelven el tensor  
26 //dichas funciones utilizan tf.variables que son los parametros entrenables  
27 function model(xs) {  
28     const hiddenLayer = tf.tidy( function(){  
29         //peso,bias y funcion RELU  
30         return xs.matMul(pesosCapaUno).add(biasCapaUno).relu();  
31     });  
32     //pesos, bias y funcion Sigmoide  
33     retmodel = hiddenLayer.matMul(pesosCapaDos).add(biasCapaDos).sigmoid();  
34     return retmodel;  
35 }  
36 //Gradientes de descenso de 0.1  
37 const optimizador = rf.train.sgd(TASA_DE_APRENDIZAJE);  
38   
39 //inicializacion aleatoria de los pesos  
40 function inicializacion(shape, prevLayerSize) {  
41     return tf.randomNormal(shape).mul(tf.scalar(Math.sqrt(2.0 / prevLayerSize)));  
42 }  
43   
44 //Creamos la funcion de costos (aunque tf tambien nos provee de varias)  
45 //aqui usamos minimos cuadrados  
46 function calculaCosto(y,output) {  
47     return tf.squaredDifference(y,output).sum().sqrt();  
48 }
```

Finalmente es aquí donde está el proceso iterativo donde vamos a intentar minimizar ese costo, le estamos pidiendo que nos regrese información cada 100 ciclos y nos desplegara en la pantalla cuál es esa minimización.

Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda nnTF.js - Visual Studio Code

El modo restringido está destinado a la navegación segura por el código. Confié en esta ventana para habilitar todas las funciones. [Administrar](#) [Más información](#)

EXPLORADOR ... nnTF.js

> NO HAY NINGUNA CARPE... C:\Users > Dell > OneDrive > Escritorio > septimo > lenguajes II > video9 > video_9 > nnTF.js > ...

ESQUEMA

- NUM_NEURONAS...
- TASA_DE_APRENDI...
- noHeParadoAntes
- ciclosDeAprendizaje
- datosEntrada
- datosEsperados
- tensorEntrada
- tensorEsperado
- pesosCapaUno
- biasCapaUno
- pesosCapaDos
- biasCapaDos
- model
- hiddenLayer
 - tf.tidy() callback
- optimizador
- inicializacion
- calculaCosto

```
//función de entrenamiento que de manera repetitiva optimiza los parámetros de la función de costo
async function entrena(iteraciones) {
  const regresaCosto =true;
  let costo;
  for(let i =0; i< iteraciones; i++){
    costo = optimizador.minimize(function() {
      return calculaCosto(tensorEsperado, model(tensorEntrada));
    },regresaCosto);
    if(i%100 === 0){
      costods = costo.dataSync()
      document.getElementById('divEntrenamiento').innerHTML += 'Pérdida['+i+']:';
      updateCiclosDeAprendizaje(ciclosDeAprendizaje);
      if(costods<0.6 && noHeParadoAntes){
        noHeParadoAntes=false;
        break;
      }
    }
  }
  await tf.nextFrame();
  ciclosDeAprendizaje += 1;
}
```

Aprendizaje de Máquina

Ciclos de Aprendizaje: 675

Oprime el botón para entrenar la red neuronal. Selecciona una operación para probar

Entrenando

Número de Iteraciones de entrenamiento (aleatorio): 451

Perdida[0]: 0.9818071126937866
Perdida[100]: 0.5134316682815552

Perdida: 0.5134316682815552

Duración del entrenamiento : 2.00 seconds

Número de Iteraciones de entrenamiento (aleatorio): 575

Perdida[0]: 0.5067521929740906
Perdida[100]: 0.20991221070289612
Perdida[200]: 0.11284975707530975
Perdida[300]: 0.07138248533010483
Perdida[400]: 0.05136469751596451
Perdida[500]: 0.03967222943902016

Perdida: 0.033911604434251785

Número de Iteraciones de entrenamiento (aleatorio): 441

Perdida[0]: 0.03378685936331749
Perdida[100]: 0.028119344264268875
Perdida[200]: 0.023982059210538864
Perdida[300]: 0.02082858234643936
Perdida[400]: 0.018601998686790466

Perdida: 0.01774444244801998

Duración del entrenamiento : 7.00 seconds

Número de Iteraciones de entrenamiento (aleatorio): 566

Perdida[0]: 0.01769555463433266
Perdida[100]: 0.015832463279366493
Perdida[200]: 0.014451807364821434
Perdida[300]: 0.013165926560759544
Perdida[400]: 0.012198948301374912
Perdida[500]: 0.011247850954532623

Perdida: 0.010735261254012585

Duración del entrenamiento : 9.00 seconds

ArchivoEditarSelecciónVerIrEjecutarTerminalAyuda

nnTF.js - Visual Studio Code

El modo restringido está destinado a la navegación segura por el código. Confíe en esta ventana para habilitar todas las funciones. Administrar Más información

EXPLORADORnnTF.js

NO HAY NINGUNA CARPETA SELECCIONADA

ESQUEMA

NUM_NEURONAS...TASA_DE_APRENDI...noHeParadoAntes

ciclosDeAprendizaje

datosEntrada

datosEsperados

tensorEntrada

tensorEsperado

pesosCapaUno

biasCapaUno

pesosCapaDos

biasCapaDos

model

hiddenLayer

tf.tidy() callback

optimizador

inicializacion

calculaCosto

entrena

regresaCosto

costo

i

optimizador.mini...

updateCiclosDeAp...

aprendeXor

timeStart

iteraciones

loss

time

pruebaXor

Modo restringido 0 0 0

Lin. 23, col. 1 Espacios: 4 UTF-8 CRLF {} JavaScript

```
69 }
70
71 updateCiclosDeAprendizaje(ciclosDeAprendizaje);
72 return costo.dataSync();
73 }
74 function updateCiclosDeAprendizaje(ciclos){
75   $("#divCiclosDeAprendizaje")[0].innerHTML="ciclos de Aprendizaje:" + ciclos;
76 }
77
78 async function aprendeXor() {
79   const timeStart = performance.now();
80   const iteraciones = Math.floor(Math.random()*200+400);
81   document.getElementById('divEntrenamiento').innerHTML += <br>Número de iteraciones<br>;
82   const loss = await entrena(iteraciones);
83   const time = performance.now()- timeStart;
84   document.getElementById('divEntrenamiento').innerHTML += <br>perdida: +loss[0]+<br>;
85   document.getElementById('divEntrenamiento').innerHTML += 'duracion del entrenamiento';
86 }
87 async function pruebaXor() {
88   var timeSart2 = 0
89   var time2 = 0
90   var strresult = "";
91   timeSart2 = performance.now();
92   for (i=0; i=datosEntrada.length; i++){
93     const inputData = tf.tensor2d([datosEntrada[i]], [1,2]);
94     const expectedOutput = tf.tensor1d(datosEsperados[i]);
95     const val = model(inputData);
96     const myVal = await val.data()
97     strresult+=strresult + "(" + datosEntrada[i][0]+","+ datosEntrada[i][0];
98   }
99   time2 = performance.now()-timeSart2;
100   document.getElementById('divPrueba').innerHTML = <br>Duracion de la prueba: +timeSart2;
101   document.getElementById('divPrueba').innerHTML += strresult;
102   document.getElementById('divPrueba').innerHTML += <br>Error: + calculaCosto(timeSart2);
103 }
104 function imprimePesosYBias(){
105   console.log(pesosCapaUno.dataSync());
106 }
```

ArchivoEditarSelecciónVerIrEjecutarTerminalAyuda

nnTF.js - Visual Studio Code

El modo restringido está destinado a la navegación segura por el código. Confíe en esta ventana para habilitar todas las funciones. Administrar Más información

EXPLORADORnnTF.js

NO HAY NINGUNA CARPETA SELECCIONADA

ESQUEMA

NUM_NEURONAS...TASA_DE_APRENDI...noHeParadoAntes

ciclosDeAprendizaje

datosEntrada

datosEsperados

tensorEntrada

tensorEsperado

pesosCapaUno

biasCapaUno

pesosCapaDos

biasCapaDos

model

hiddenLayer

tf.tidy() callback

optimizador

inicializacion

calculaCosto

entrena

regresaCosto

costo

i

optimizador.mini...

updateCiclosDeAp...

aprendeXor

timeStart

iteraciones

loss

time

pruebaXor

Modo restringido 0 0 0

Lin. 23, col. 1 Espacios: 4 UTF-8 CRLF {} JavaScript

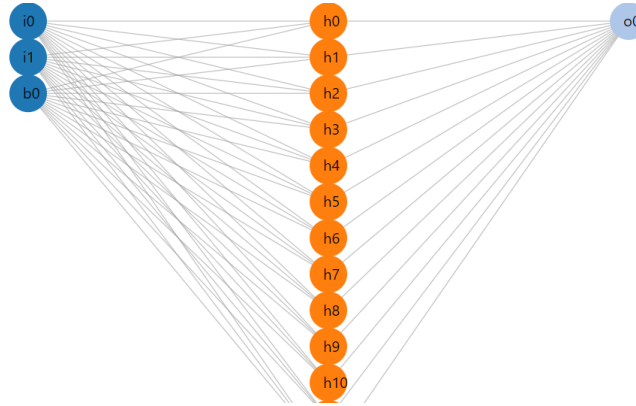
```
83 const time = performance.now()- timeStart;
84 document.getElementById('divEntrenamiento').innerHTML += <br>perdida: +loss[0]+<br>;
85 document.getElementById('divEntrenamiento').innerHTML += 'duracion del entrenamiento';
86 }
87 async function pruebaXor() {
88   var timeSart2 = 0
89   var time2 = 0
90   var strresult = "";
91   timeSart2 = performance.now();
92   for (i=0; i=datosEntrada.length; i++){
93     const inputData = tf.tensor2d([datosEntrada[i]], [1,2]);
94     const expectedOutput = tf.tensor1d(datosEsperados[i]);
95     const val = model(inputData);
96     const myVal = await val.data()
97     strresult+=strresult + "(" + datosEntrada[i][0]+","+ datosEntrada[i][0];
98   }
99   time2 = performance.now()-timeSart2;
100   document.getElementById('divPrueba').innerHTML = <br>Duracion de la prueba: +timeSart2;
101   document.getElementById('divPrueba').innerHTML += strresult;
102   document.getElementById('divPrueba').innerHTML += <br>Error: + calculaCosto(timeSart2);
103 }
104 function imprimePesosYBias(){
105   console.log(pesosCapaUno.dataSync());
106   console.log(biasCapaUno.dataSync());
107   console.log(pesosCapaDos.dataSync());
108   console.log(biasCapaUno.dataSync());
109 }
```

Probando: pruebaXor()

Duración de la prueba : 52.900 milisegundos

(0 , 0)-->0.009 (esperado: 0)
(0 , 1)-->0.997 (esperado: 1)
(1 , 0)-->0.997 (esperado: 1)
(1 , 1)-->0.003 (esperado: 0)

Error: Tensor 0.01080295629799366



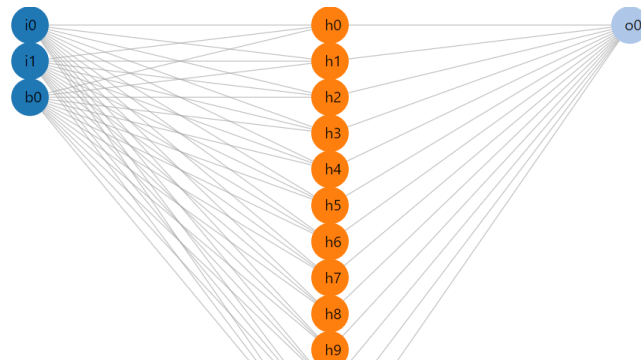
Duración del entrenamiento : 9.00 seconds

Probando: pruebaXor()

Duración de la prueba : 52.900 milisegundos

(0 , 0)-->0.009 (esperado: 0)
(0 , 1)-->0.997 (esperado: 1)
(1 , 0)-->0.997 (esperado: 1)
(1 , 1)-->0.003 (esperado: 0)

Error: Tensor 0.01080295629799366



DevTools is now available in Spanish!

Always match Chrome's language Switch DevTools to Spanish Don't show again

Elements Console Sources Network

```
<g transform="translate(160,19.23076923076923)">...</g>
<g transform="translate(160,57.69230769230769)">...</g>
<g transform="translate(160,96.15384615384615)">...</g>
<g transform="translate(800,19.23076923076923)">...</g>
<g transform="translate(480,19.23076923076923)">...</g>
<g transform="translate(480,57.69230769230769)">...</g>
<g transform="translate(480,96.15384615384615)">...</g>
<g transform="translate(480,134.6153846153846)">...</g>
<g transform="translate(480,173.07692307692307)">...</g>
<g transform="translate(480,211.53846153846152)">...</g>
<circle class="node" r="20" style="fill: rgb(255, 127, 14);"></circle>
== $0
<text dx="-.35em" dy="-.35em">h5</text>
</g>
<g transform="translate(480,250)">...</g>
<g transform="translate(480,288.46153846153845)">...</g>
```

jQuery341099385597789200931: {}

__data__: {label: 'h5', layer: 2, lidx: 6, x: 480, y: 211.53846153846152}

ATTRIBUTE_NODE: 2
CDATA_SECTION_NODE: 4
COMMENT_NODE: 8
DOCUMENT_FRAGMENT_NODE: 11
DOCUMENT_NODE: 9
DOCUMENT_POSITION_CONTAINED_BY: 16
DOCUMENT_POSITION_CONTAINS: 8
DOCUMENT_POSITION_DISCONNECTED: 1
DOCUMENT_POSITION_FOLLOWING: 4
DOCUMENT_POSITION_IMPLEMENTATION_SPECIFIC: 32
DOCUMENT_POSITION_PRECEDING: 2
DOCUMENT_TYPE_NODE: 10
ELEMENT_NODE: 1
ENTITY_NODE: 6
ENTITY_REFERENCE_NODE: 5
NOTATION_NODE: 12

Duración del entrenamiento : 9.00 seconds

Probando: pruebaXor()

Duración de la prueba : 52.900 milisegundos

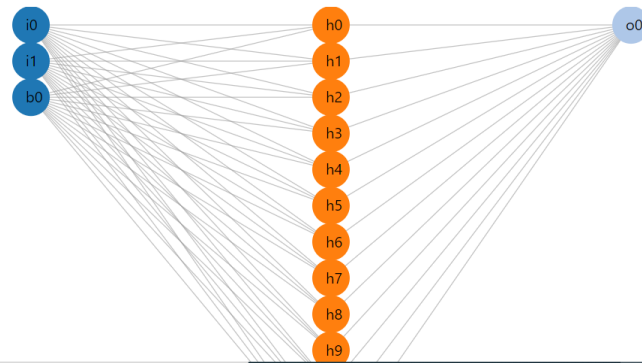
(0 , 0)-->0.009 (esperado: 0)

(0 , 1)-->0.997 (esperado: 1)

(1 , 0)-->0.997 (esperado: 1)

(1 , 1)-->0.003 (esperado: 0)

Error: Tensor 0.01080295629799366



DevTools is now available in Spanish!

Always match Chrome's language Switch DevTools to Spanish Don't show again

Elements Console Sources Network

```
<g transform="translate(480,134.6153846153846)">...</g>
<g transform="translate(480,173.07692307692307)">...</g>
<g transform="translate(480,211.53846153846152)">
  <circle class="node" r="20" style="fill: rgb(255, 127, 14);"></circle>
  == $0
  <text dx="-.35em" dy="-.35em">h5</text>
</g>
<g transform="translate(480,250)">...</g>
<g transform="translate(480,288.46153846153845)">...</g>
<g transform="translate(480,326.9230769230769)">...</g>
<g transform="translate(480,365.38461538461536)">...</g>
<g transform="translate(480,403.8461538461538)">...</g>
<g transform="translate(480,442.30769230769226)">...</g>
<g transform="translate(480,480.7692307692308)">...</g>
</svg>
</body>
</html>
```

ter.csscolumns-breakinside.flexbox.picture.srcset.webworkers body svg g circle.node

Styles Computed Layout Event Listeners DOM Breakpoints Properties Accessibility

jQuery341099385597789200931: {}

__data__: {label: 'h5', layer: 2, lid: 6, x: 480, y: 211.53846153846152}

ATTRIBUTE_NODE: 2

CDATA_SECTION_NODE: 4

COMMENT_NODE: 8

DOCUMENT_FRAGMENT_NODE: 11

DOCUMENT_NODE: 9

DOCUMENT_POSITION_CONTAINED_BY: 16

DOCUMENT_POSITION_CONTAINS: 8

DOCUMENT_POSITION_DISCONNECTED: 1

DOCUMENT_POSITION_FOLLOWING: 4

DOCUMENT_POSITION_IMPLEMENTATION_SPECIFIC: 32

DOCUMENT_POSITION_PRECEDING: 2

DOCUMENT_TYPE_NODE: 10

ELEMENT_NODE: 1

ENTITY_NODE: 6

ENTITY_REFERENCE_NODE: 5

NOTATION_NODE: 12