27-3-2020

Mi primera web con redes neuronales

Inteligencia artificial

José Alejandro Musito García
INSTITUTO TECNOLOGICO DE CUAUTLA

Contenido

Index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="ia.css">
    <script src="//unpkg.com/brain.js"></script>
    <script src="ia.js"></script>
</head>

<body onload="iniciarJuego()">
</body>

</html>
```

Es un archivo HTML común, donde agregamos los enlaces para el archivo css, y los archivos JavaScript, incluyendo la librería bran.js. En la etiqueta body le adjuntamos la función iniciarJuego cuando cargue, esta función se explica en el archivo JS, pero, a grandes rasgos, inicializa todos los componentes necesarios.

la.css

```
canvas {
    border: 1px solid #000000;
}
```

Su única función es agregarle el borde al elemento Canvas que se realiza en el archivo ia.js

la.js

Inicializamos la variable network que representa nuestra red neuronal y la entrenamos con los datos correspondientes de la ubicación del tiro al blanco resultante. En la siguiente tabla y formulas observaremos mejor la relación de estos números.

$$x = \frac{(xm - xp)}{xm}$$

$$y = \frac{(ym - yp)}{ym}$$

Donde $ym = y\ mas\ grande$, $yp = y\ circulo\ pequeño$, $xm = x\ mas\ grande$ y $xp = x\ circulo\ pequeño$

X circulo pequeño	Y circulo pequeño	X	Y
230	360	1	0.5
250	360	0	0.5
240	350	0.5	1
240	370	0.5	0
140	360	0.5	0.5

$$xg = x * 100$$

$$yg = 620 + (y * 100)$$

Donde $xg = x\ circulo\ grande$ y $yg = y\ circulo\ grande$

X	Y	X circulo grande	Y circulo grande
0	0.5	0	670
1	0.5	100	670
0.5	0	50	620
0.5	1	50	720
0.5	0.5	50	670

Cabe aclarar que se adjuntaron las formulas en esta parte para aclarar las variables, sin embargo, la red neuronal solamente recibe y otorga valores del 0 al 1, por esto fue que las coordenadas se tuvieron que tratar antes y después.

```
const network = new brain.NeuralNetwork();
console.log(
    network.train(
        [
```

```
            { input: { x: 1, y: 0.5 }, output: { x: 0, y: 0.5 } },
            { input: { x: 0, y: 0.5 }, output: { x: 1, y: 0.5 } },
            { input: { x: 0.5, y: 1 }, output: { x: 0.5, y: 0 } },
            { input: { x: 0.5, y: 0 }, output: { x: 0.5, y: 1 } },
            { input: { x: 0.5, y: 0.5 }, output: { x: 0.5, y: 0.5 } }
        ]
    )
);
var bolita;
```

Realizamos nuestro elemento Canvas y lo insertamos en el HTML, este elemento realiza todos los gráficos a mostrar. La iniciamos como un objeto con los atributos Canvas, iniciar y limpiar, donde los dos últimos actúan como una función de la variable. La primera crea el Canvas y lo adjunta en el HTML, y posteriormente dibuja los graficas. La segunda función simplemente borra y crea nuevamente los gráficos para mostrar el nuevo punto de toque.

```
var areaCanvas = {
    canvas: document.createElement('canvas'),
    iniciar: function () {
        this.canvas.width = 480;
        this.canvas.height = 720;
        this.context = this.canvas.getContext("2d");
        document.body.insertBefore(this.canvas, document.body.childNodes[0])
;
        hacerFondo();
    },
    limpiar: function () {
        this.context.clearRect(0, 0, this.canvas.width, this.canvas.height);
        hacerFondo();
    }
}
```

En la creación de nuestro elemento Canvas, hacemos llamado a la función hacerFondo, la cual crearan los gráficos mostrados en la página, primero se crea el rectángulo para el cielo en nuestro juego, después el rectángulo para el pasto, y por ultimo los dos blancos de tiros que son un conjunto de círculos realizados con ciclos for. En esos ciclos ingresamos las coordenadas iniciales que usamos en las tablas de arriba.

```
function hacerFondo() {
    contexto = areaCanvas.context;
    contexto.beginPath();
    contexto.fillStyle = 'skyblue';
```

```
    contexto.fillRect(0, 0, 480, 360);
    contexto.fill();

    contexto.beginPath();
    contexto.fillStyle = 'green';
    contexto.fillRect(0, 360, 480, 360);
    contexto.fill();

    radio = 10;
    for (let i = 0; radio > 0; i++) {
        contexto.beginPath();
        if (i % 2 != 0)
            contexto.fillStyle = 'white';
        else
            contexto.fillStyle = 'red';
        contexto.arc(240, 360, radio, 0, 2 * Math.PI);
        contexto.fill();
        contexto.stroke();
        radio -= 2;
    }

    radio = 50;
    for (let i = 0; radio > 0; i++) {
        contexto.beginPath();
        if (i % 2 != 0)
            contexto.fillStyle = 'white';
        else
            contexto.fillStyle = 'red';
        contexto.arc(50, 670, radio, 0, 2 * Math.PI);
        contexto.fill();
        contexto.stroke();
        radio -= 10;
    }
}
```

Creamos el constructor para el circulo a mostrar dentro del blanco de tiro mas grande. Sus atributos son los básicos para dibujar un círculo en HTML, además de la función actualizar, donde se dibuja el circulo.

```
function bolita(radio, x, y) {
    this.radio = radio;
    this.x = x;
    this.y = y;
    this.actualizar = function () {
        contexto = areaCanvas.context;
        contexto.beginPath();
```

```
        contexto.fillStyle = 'blue';
        contexto.arc(this.x, this.y, this.radio, 0, 2 * Math.PI);
        contexto.fill();
        contexto.stroke();
    }
}
```

Con los dos objetos necesarios para nuestro juego, creamos la función actualizarArea, la cual simplemente hará llamado a los métodos correspondientes de los objetos para que se dibuje nuevamente todos los graficos del Canvas.

```
function actualizarArea() {
    areaCanvas.limpiar();
    bolita.actualizar();
}
```

Ahora creamos la función procesar, la cual será la encargada del procesamiento a hacer cuando se haga click en algún lugar del blanco de tiro mas pequeño. En esta función es donde se usa el método run de nuestra red neuronal y nos devolverá el procesamiento que realizo, obteniendo el resultado graficamos en el círculo más grande.

```
function procesar(x, y) {
    contexto = areaCanvas.context;
    contexto.beginPath();
    contexto.arc(240, 360, 10, 0, 2 * Math.PI);
    contexto.closePath();

    if (contexto.isPointInPath(x, y)) {
        var entrada = {
            x: (250 - x) / 20,
            y: (370 - y) / 20
        }

        console.log(entrada);

        var resultado = network.run(entrada);

        console.log(resultado['x'] + '|' + resultado['y']);
        bolita.x = (resultado['x'] * 100);
        bolita.y = 620 + (resultado['y'] * 100);
        actualizarArea();
    }
}
```

Finalizamos con la función que llamamos en el body del HTML, donde inicializamos todos los objetos con las funciones correspondientes.

```
function iniciarJuego() {
    bolita = new bolita(5, 50, 670);
    areaCanvas.iniciar();

    areaCanvas.canvas.onclick = function (e) {
        var rect = this.getBoundingClientRect(),
            x = e.clientX - rect.left,
            y = e.clientY - rect.top;
        procesar(x, y);
    }
}
```