Inteligencia Artificial

Practica 3: Introduccion a PyTorch

Profesor: Enrique Francisco Soto Astorga
Ayudante de Laboratorio: Ricardo Enrique Pérez Villanueva

March 14, 2025

1 Proyecto

Vamos a implementar una especie de modelo parecido a un Perceptrón[1] usando la biblioteca para redes neuronales (y más) para Python, llamado **PyTorch**. Vamos a crear una red neuronal que haga la predicción de un Autómata el cual representa un Counter[2] y un bit que representa una entrada[3]. El valor de la entrada cambia el estado del autómata, haciendo que el contador avance o se detenga.
Vamos a crear una red neuronal de 3 inputs y 2 outputs, la cual haga una predicción del resultado de el contador, dado en que numero se encuentra ahora y que entrada esta recibiendo. El autómata esta descrito en la siguiente diagrama:

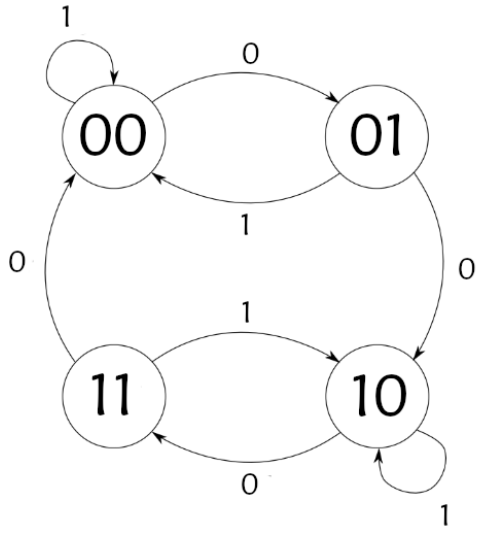

Figure 1: Autómata que describe el funcionamiento de un counter. Notemos como cambia de 0, 1, 2 y 3 en binario si las entradas son 0, y si son 1, detiene el counter o hace que se regrese a un estado anterior.

2 Implementación

2.1 Redes Neuronales en PyTorch

Vamos a implementar una red neuronal usando PyTorch. PyTorch es un framework para realizar, entre muchas cosas, redes neuronales. La implementación de una red neuronal, el modelo en el que esta basado, las funciones de perdida, optimización y el entrenamiento se pueden realizar en menos de 50 lineas de código. Vamos a realizar una red neuronal con 3 inputs y 2 outputs, para predecir los estados de nuestro autómata (más adelante analizamos como interpretar los datos del

[1] No es realmente un precpetron porque las funciones de activacion y el algoritmo son diferentes, pero son parecidos.

[2] Un Counter es una operación en Lógica Digital, el cual genera números binarios en orden. Es decir, cuenta de 0, 1, 2 , 3 ... ect, pero en binario.

[3] Para no generar confusiones, al input del autómata, se le llamara Entrada, y a los inputs de las redes neuronales, se les llamara inputs por el resto de la practica.

autómata para introducirlos en esta red neuronal). A continuación, hago un pequeño desglose de como funcionan las redes neuronales en PyTorch.

2.1.1 torch.nn

Para empezar a usar redes neuronales en PyTorch, necesitamos la biblioteca "torch.nn" y crear un modelo. Los modelos se crean como clases que reciben como parámetro un objeto de tipo **nn.Module**.

- Este objeto tiene 2 funciones importantes, la función constructora donde se debe hacer una llamada a la clase padre y una función llamada **forward**. La función forward representa la parte de Forward Propagation, es decir, la función de activación.
- Después tenemos que escoger una función de perdida y una función de optimización. Las funciones de perdida esta en la biblioteca torch, y para poder usar las funciones de optimización, debemos importar la biblioeta **torch.optim**.
- Después tenemos el paso de aprendizaje, aquí es donde vamos a generar predicciones usando el modelo del perceptron, calculamos el valor de la función de perdida, usamos el optimizador para minimizar la perdida y realizamos la **Backward Propagation** con la función **backward**, de esta forma corrigiendo los errores cometidos.
- Finalmente se prueba el modelo con datos de entrada, en este caso, vamos a usar los mismos datos con los cuales entrenamos el modelo para probar la precisión del modelo.

Para la red neuronal de esta practica, vamos a usar una **función sigmoide** como función de activación, como función de perdida usaremos **Binary Cross Entropy** y como optimizador, usaremos **Stochastic gradient descent**. Las epocas y el Learning Rate se deja a su decision.

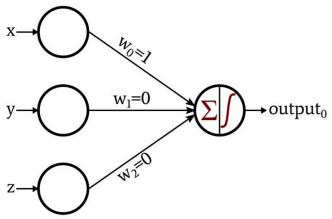

Figure 2: Modelo que representa un perceptron.

2.2 Autómata

El autómata describe un contador que recibe una entrada y dada esa entrada, decide si mantenerse en el numero actual o cambiar de numero, notablemente, cuando la entrada es 0, el numero siempre cambia, y cuando la entrada es 1, el numero se mantiene o se regresa al numero anterior. El comportamiento de la red de neural esta descrito en la siguiente tabla de verdad:

Estado Actual		Ent-	Estado Sig. Dado la Entrada	
0	0	1	0	0
0	0	0	0	1
0	1	1	0	0
0	1	0	1	0
1	0	1	1	0
1	0	0	1	1
1	1	1	1	0
1	1	0	0	0

Figure 3: Tabla de estados, notese que que los estados son 2 bits y la entrada es solo 1.

De esta forma, vamos a representar los estados del autómata y su entrada en nuestra red neuronal como un 3 números dentro de un arreglo. Por ejemplo, cuando estamos en el estado 00 y tenemos como entrada el 0, representamos este dato como un [0,0,0] dentro del tensor de datos en PyTorch. Finalmente, podemos representar el estado al que se llego de la misma forma, es decir, un arreglo de 2 elementos. Por ejemplo, el estado al que llega el arreglo [0,0,0] sería [0,0]. De esta forma, ya tenemos nuestros datos "inputs" o "X" y las "labels" o "Y", respectivamente.
En total, se deben tener 8 inputs de 3 bits, y 8 outputs de 2 bits.

3 Entrega

Se va a calificar que la red neuronal funcione bien, y tenga muy pocos errores. Se va a evaluar que la función de perdida al final del entrenamiento tenga un valor de 0.2 o menor, es decir, que su modelo sea preciso.

- Se recomienda que se haga mediante un Notebook en Google Collab
- Se recomienda que utilicen la implementación que se vio en clase, es decir, compartir con un Link en el Classroom su archivo de Google Collab donde hicieron la Red Neuronal, recuerden dar permisos de ejecución.
- Se va a calificar la precisión del modelo, como se menciona arriba, que la función de perdida al final tenga un valor de 0.2 o menor es un buen comienzo, aquí es donde ustedes deben modificar la Learning Rate y las épocas de entrenamiento en su modelo.
- Se va a calificar que el modelo use los modelos descritos arriba (Sigmoid Function, Binary Cross Entropy y Stochastic Gradient Descent).
- Tambien se va a evaluar que, en la parte del entrenamiento comenten de forma detallada que hace cada linea de código (básicamente, que me cuenten donde ocurre el backward propagation, la optimización, el calculo de la función de perdida, ect.)

4 Documentación y materiales útiles:

A continuación se les ofrece fuentes de información que les van a ayudar a realizar su practica

- Documentación de Module en Pytorch `https://pytorch.org/docs/stable/generated/torch.nn.Module.html`
- Información sobre Stochastic Gradient Descent: `https://en.wikipedia.org/wiki/Stochastic_gradient_descent`
- Información sobre Binary Cross Entropy: `https://www.geeksforgeeks.org/binary-cross-entropy-log-los`
- Información sobre la funcion Sigmoide: `https://en.wikipedia.org/wiki/Sigmoid_function`
- Información sobre el Counter: `http://geeksforgeeks.org/counters-in-digital-logic/`