Analisis juego Cartas de la memoria

Juan ALejandro Diaz Lote
Pontificia Universidad Javeriana
25 de noviembre de 2022

1. Introducción

Para el desarrollo de aplicaciones a nivel profesional es muy importante el manejo de los recursos de harware en los cuales el codigo se va a ejecutar. Es aqui donde entre el analisis de algoritmos, porque con ayuda de este se pueden hacer estimaciones teóricas de los recursos que necesita. Usualmente, los recursos a los cuales se hace referencia son el tiempo y el almacenamiento. Con estas estimaciones se puede determinar si los metodos usados en la programacion son los mas optimos, debido a que un problema se puede resolver con multiples algoritmos pero siempre abra uno mas optimo que otro. En python se pueden encontar varias librerias para hacer varias operaciones matematicas. recursivas, graficas que potencian el juego y lo hacen mas eficiente. Debido a esto se buscara analizar el codigo fuente de un juego sencillo como lo es el juego cartas de memoria para analizar el uso de estos metodos que nos ofrecen las liberias.

Figura 1: Cartas de la memoria

2. Objetivos

2.1. Objetivo general:

Observar como funciona el reconomiento de imgenes a traves de machine learning

2.2. Objetivos especificos:

1. Recononer la complejidad de algunas funciones dentro del codigo las cuales hacen que este funcione de manera correcta
2. Probar que sucede con los tiempos de ejecucuion si se utilizan los diferentes algoritmoscon varias cantidades de datos(en este caso los datos considerados como el numero de cartas en el juego)
3. Realizar una red neuronal que pueda resolver el juego de memoria de cartas
4. Resolver a traves de un algoritmo de machine learning usando el reconocimineto de imagenes

3. Marco teorico

Pygame es un conjunto de módulos de Python diseñados para escribir videojuegos. Pygame agrega funcionalidad además de la excelente biblioteca SDL. Esto le permite crear juegos y programas multimedia con todas las funciones en el lenguaje python. Pygame es altamente portátil y se ejecuta en casi todas las plataformas y sistemas operativos.Pygame es gratis. Lanzado bajo la licencia LGPL, puede crear juegos de código abierto, freeware, shareware y comerciales con él.

Las CPU multinúcleo se pueden usar fácilmente. Con CPU de doble núcleo común y CPU de 8 núcleos disponibles a bajo precio en los sistemas de escritorio, el uso de CPU de varios núcleos le permite hacer más en su juego. Las funciones de pygame seleccionadas liberan el temido Python GIL, que es algo que puede hacer desde el código C.

Utiliza código C y ensamblador optimizado para las funciones principales. El código C suele ser de 10 a 20 veces más rápido que el código de Python, y el código ensamblador puede ser fácilmente 100 veces o más rápido que el código de Python.

Se han publicado muchos juegos. Incluyendo finalistas del Indie Game Festival, finalistas del Australian Game festival, shareware popular, proyectos multimedia y juegos de código abierto. Se han publicado más de 660 proyectos en los sitios web de pygame, tales como: lista necesaria. Se han lanzado muchos más juegos con SDL (en el que se basa pygame), por lo que puede estar seguro de que muchos de ellos han sido bien probados por millones de usuarios.

Figura 2: ejemplos hechos con pygame

Ahora debemos decir que la recursividad no es una estructura de datos, sino que es una técnica de programación que nos permite que un bloque de instrucciones se ejecute n veces. Remplaza en ocasiones a estructuras repetitivas.

La recursividad es un concepto difícil de entender en principio, pero luego de analizar diferentes problemas aparecen puntos comunes. En Java los métodos pueden llamarse a sí mismos. Si dentro de un método existe la llamada a sí mismo decimos que el método es recursivo. Cuando un método se llama a sí mismo, se asigna espacio en la pila para las nuevas variables locales y parámetros. Al volver de una llamada recursiva, se recuperan de la pila las variables locales y los parámetros antiguos y la ejecución se reanuda en el punto de la llamada al método.

En informática, la programación dinámica es un método para reducir el tiempo de ejecución de un algoritmo mediante la utilización de subproblemas superpuestos y subestructuras óptimas, como se describe a continuación. El matemático Richard Bellman inventó la programación dinámica en 1953 que se utiliza para optimizar problemas complejos que pueden ser discretizados y secuencializados.

Una subestructura óptima significa que se pueden usar soluciones óptimas de subproblemas para encontrar la solución óptima del problema en su conjunto. Por ejemplo, el camino más corto entre dos vértices de un grafo se puede encontrar calculando primero el camino más corto al objetivo desde todos los vértices adyacentes al de partida, y después usando estas soluciones para elegir el mejor camino de todos ellos. En general, se pueden resolver problemas con subestructuras óptimas siguiendo estos tres pasos:

1. Dividir el problema en subproblemas más pequeños.

2. Resolver estos problemas de manera óptima usando este proceso de tres pasos recursivamente.

3. Usar estas soluciones óptimas para construir una solución óptima al problema original

.

Los subproblemas se resuelven a su vez dividiéndolos en subproblemas más pequeños hasta que se alcance el caso fácil, donde la solución al problema es trivial.

Tamebien debemos hablar de machine learning, El aprendizaje automático es una forma de inteligencia artificial que permite que un sistema aprenda de los datos en lugar de aprender mediante programación explícita. Sin embargo, el aprendizaje automático no es un proceso simple. A medida que el algoritmo recopila datos de entrenamiento, es posible producir modelos más precisos basados en los datos. Un modelo de aprendizaje automático es la salida de datos que se crea cuando entrena un algoritmo de aprendizaje automático en datos. Después del entrenamiento, el modelo recibe una entrada y produce una salida. Por ejemplo, un algoritmo predictivo crea un modelo predictivo. Luego, cuando alimenta datos a un modelo predictivo, obtiene una predicción basada en los datos que entrenaron el modelo.

Donde se usara un tipo de machine learning que lleva por nombre Deep learning. El aprendizaje profundo es un método especial de aprendizaje automático que combina redes neuronales con capas secuenciales para aprender iterativamente de los datos. El aprendizaje profundo es particularmente útil cuando se aprenden patrones a partir de datos no estructurados. Las complejas redes neuronales de aprendizaje profundo están diseñadas para imitar la forma en que funciona el cerebro humano, de modo que se pueda enseñar a las computadoras a lidiar con abstracciones y problemas mal definidos. Las redes neuronales y el aprendizaje profundo se utilizan a menudo en aplicaciones de reconocimiento de imágenes, voz y visión artificial.

4. Desarrollo

Primero se hace la interfaz del juego para esto se debe utilizar a traves de la libreria pygame, esto debido a que esta permite tomar los datos de la pantalla como los clicks, las dimensiones de la pantalla, etc.

En el siguiente se muestra como con esta libreria se inicializa el juego que despliega una pantalla teniendo en cuenta el tamaño de la panatalla del computador, ademas de esto se crean los botones para ver las fichas y ademas poder dar click a iniciar el juego. Tambien haciendo uso de efectos como musica ambiental y sonidos para cuando el jugador da click sobre las cartas.

```
pantalla_juego = pygame.display.set_mode((
    anchura_pantalla, altura_pantalla))
pygame.display.set_caption('Memorama en Python -
    By Parzibyte')
pygame.mixer.Sound.play(sonido_fondo, -1)  # El
    -1 indica un loop infinito
# Ciclo infinito...
while True:

    for event in pygame.event.get():
        # Si quitan el juego, salimos
        if event.type == pygame.QUIT:
            sys.exit()
        # Si hicieron clic y el usuario puede
            jugar...
```

```
        elif event.type == pygame.
            MOUSEBUTTONDOWN and puede_jugar:


            xAbsoluto, yAbsoluto = event.pos
            if boton.collidepoint(event.pos):
                if not juego_iniciado:
                    iniciar_juego()

            else:
                # Si no hay juego iniciado,
                    ignoramos el clic
                if not juego_iniciado:
                    continue

                x = math.floor(xAbsoluto /
                    medida_cuadro)
                y = math.floor(yAbsoluto /
                    medida_cuadro)

                cuadro = cuadros[y][x]
                if cuadro.mostrar or cuadro.
                    descubierto:
                    # continue ignora lo de
                        abajo y deja que el
                        ciclo siga
                    continue

                if x1 is None and y1 is None:
                    # Entonces la actual es en
                        la que acaban de dar
                        clic, la mostramos
                    x1 = x
                    y1 = y
                    cuadros[y1][x1].mostrar =
                        True
                    pygame.mixer.Sound.play(
                        sonido_voltear)
                else:
                    # En caso de que ya hubiera
                        una clickeada
                        anteriormente y estemos
                        buscando el par,
                        comparamos...
                    x2 = x
                    y2 = y



                    cuadros[y2][x2].mostrar =
                        True
                    cuadro1 = cuadros[y1][x1]
                    cuadro2 = cuadros[y2][x2]
                    # Si coinciden, entonces a
                        ambas las ponemos en
                        descubiertas:


                    if cuadro1.fuente_imagen ==
                        cuadro2.fuente_imagen:
                        cuadros[y1][x1].
                            descubierto = True
                        cuadros[y2][x2].
                            descubierto = True



                        x1 = None
                        x2 = None
                        y1 = None
                        y2 = None
                                pygame.mixer.Sound.play(
                                    sonido_clic)
                            else:
                                pygame.mixer.Sound.play(
                                    sonido_fracaso)
                                ultimos_segundos = int(
                                    time.time())
                                # Hasta que el tiempo se
                                    cumpla, el usuario
                                    no puede jugar
                                puede_jugar = False
                        comprobar_si_gana()

        ahora = int(time.time())
```

Luego de ejecutar el codigo anterior lo que el usuario veria en panatalla seria lo siguiente.

Figura 3: Interfaz del juego

Ahora con ayuda de la liberira tensorflow y con las bases de datos que ofrece se creera una red neuronal que se entrenara con imagenes de las bases de datos. Para la red neuronal se utilza la funcion de activacion relu que no acepta numeros negativos y ademas le permite resolver problemas que no sean lineales. En la ultima capa se usa sofmax debido a que como tenemos varias categorias y varias neuronas queremos que de solo una solucion que sera la que este mas cercana a 1.

```
modelo = tf.keras.Sequential([
  tf.keras.layers.Flatten(input_shape=(28,28,1))
      , #1 - blanco y negro
  tf.keras.layers.Dense(50, activation=tf.nn.
      relu),
  tf.keras.layers.Dense(50, activation=tf.nn.
      relu),
  tf.keras.layers.Dense(10, activation=tf.nn.
      softmax) #Para redes de clasificacion


])
```

Por ultimo se define el modelo que usaremos con el ptimizador adam y su funcion de perdida SparseCategoricalCrossentropy() que al igual que sofmax se usa cuando se hacen redes neuronales de clasificacion.

```
#Compilar el modelo
modelo.compile(
    optimizer='adam',
    loss=tf.keras.losses.
        SparseCategoricalCrossentropy(),
    metrics=['accuracy']
)
```

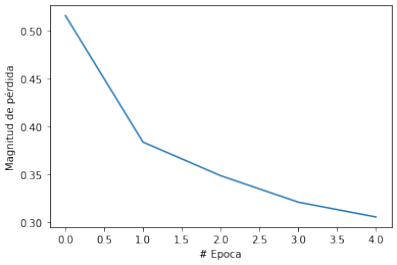

Figura 4: funcion de perdida

5. Analisis

Se puede observar al momento de entrenar la red comoo esta tomo mas tiempo que una red neuronal lineal debido a que las funciones de activacion que se estan usando son mas conplejas y permiten hacer clasificaciones en este caso de imagenes.

Para este caso lo que realiza la red neuronal es ver la poision de los pixeles de colores blanco y negro para aprenderce la posicion que ocupa cada pixel y asi luego poder predecir lo que esta en la imagen.

Esto representa un problema para la red debido a que si la imagen que recibe por entrada esta en una posicion no natural como de cabeza o en diagonal, no sera capaz de detectar que objeto hay en la imagen.

6. Conclusiones

1. El modelo tiene limitaciones al momento de que cualquier imagen que reciba dara un resultado sin importar que sea. por ejemplo si se pone un leon y la clasificacion es de solo ropa, dira que es alguna prenda.

2. La red no puede predecir el resultado correcto si recibe una imagen en una posicion poco comun es decir si la imagen de una camiseta esta de cabeza, la red no prodra reconocerla

3. Para una mayor efectividad de la red se deberia usar una red neuronal convulcional debido a que estas no aprenden de memoria sino que son capaces de interpretar lo que estan viendo

En el siguiente link usted podra encontrar un repositorio en github con todo el codigo funcional que se explica en este documento:

https://github.com/AlejoDiazLote/proyectoAnalisisAlgoritmos.git