TECNOLÓGICO NACIONAL DE MÉXICO

Inteligencia Artificial

Tarea:

Ejemplos prácticos de los enfoques

Alumno:

Rementeria Medina Jesus Hector

Enfoque conexionista (Reconocimiento Facial para desbloquear un teléfono)

Problema:

Se necesita identificar a una persona en específico mediante el reconocimiento fácil

Entrada:

Una imagen del rostro de la persona

Características faciales extraídas de imágenes previamente almacenadas

Salida:

Un valor binario (1 si se reconoció el rostro, 0 si no se pudo reconocer)

Objetivo:

Autentificar a los usuarios sin necesidad del uso de contraseñas

Beneficios:

Mayor seguridad y comodidad para los usuarios

Aprendizaje automático mejorando con el tiempo

Reconocimiento incluso con variación en iluminación y expresiones faciales

Limitaciones:

Puede fallar si la iluminación es muy pobre o el rostro se encuentra parcialmente cubierto

Requiere una gran cantidad de datos y procesamiento computacional

Puede confundirse o ser engañado por imágenes o personas muy similares por ejemplo gemelos

Enfoque Biotecnología (Optimización de ruta de drones)

Problema:

Una empresa de paquetería quiere hacer uso de drones para llevar a cabo entregas de forma eficiente minimizando el tiempo y consumo de energía

Entrada:

Ubicación destino

Restricciones geográficas (Clima, Edificios, Zonas prohibidas de vuelo)

Capacidad energética, Peso de la carga

Salida:

La ruta más óptima para minimizar la distancia y el tiempo de entrega del dron

Objetivo:

Optimizar la entrega de los paquetes mediante drones reduciendo costos y tiempos de transportes

Beneficios:

Encuentra rutas y soluciones optimas que un humano no podría calcular fácilmente

Se adapta según las condiciones climáticas o el tráfico aéreo

Limitaciones:

Los algoritmos genéticos pueden tener un costo computacional alto

No siempre garantiza la mejor solución en cada intento

Puede ser difícil de integrar junto con otros sistemas logísticos

Enfoque computacional (Predicción del precio de acciones)

Problema:

Se quiere predecir el precio de una acción en el mercado utilizando modelos de aprendizaje basados en datos históricos

Entrada:

Historial de precios de la acción a analizar

Indicadores financieros como volumen de transacciones y la tendencia en el mercado

Noticias y datos económicos que puedan influir en el valor

Salida:

La predicción más cercana posible sobre el precio de la acción según el tiempo que se quiere calcula

Objetivo:

Ayudar a los inversionistas a tomar mejores decisiones informadas sobre la compra y venta de acciones

Beneficios:

Puede procesar grandes volúmenes de datos rápidamente

Permite identificar patrones que los humanos pasarían por alto

Se ajusta constantemente con nuevos datos

Limitaciones:

No puede predecir eventos inesperados

Requiere datos de alta calidad para obtener resultados precisos

Puede generar sobreajuste si el modelo es sobrentrenado con datos pasados

Enfoque simbólico (Sistema Experto de Diagnostico medico)

Problema:

Se quiere implementar un sistema de diagnóstico automatizado para ayudar a los médicos a identificar enfermedades basadas en síntomas

Entrada:

Síntomas del paciente

Datos médicos básicos (Edad, Sexo, antecedentes clínicos)

Salida:

Diagnostico sugerido basado en las reglas del sistema

Recomendación de pruebas adicionales o tratamientos

Objetivo:

Asistir a los médicos proporcionando diagnósticos preliminares

Beneficios:

Detección rápida y eficiente de enfermedades comunes

Fácil de explicar y depurar

Limitaciones:

No puede aprender nuevos casos por sí solo, necesita actualizaciones manuales

El sistema puede ser rígido si los síntomas no coinciden exactamente según las reglas definidas

No es adecuado para diagnosticar enfermedades complejas

Ejemplo Sistema Experto (Simbólico)

El programa consiste de un sistema experto con el objetivo de simular un juego donde en base a ciertas preguntas se intenta adivinar en que personaje se está pensando

```
personaje(jonathan, humano, hombre, ley).
personaje(nozomi,humano,mujer,ley).
personaje(flynn, humano, hombre, neutral).
personaje(issabeu, humano,mujer,neutral).
personaje(walter, humano, hombre, caos).

personaje(satan,demonio,hombre,ley).
personaje(mothman, demonio, hombre, neutral).
personaje(yaskini, demonio, mujer, neutral).
personaje(demi_fiend, demonio, hombre, caos).
personaje(lilith,demonio,mujer,caos).

personaje(ace_frost, hada, hombre, ley).
personaje(jack_frost, hada, hombre, neutral).
personaje(pixie, hada, mujer, neutral).
personaje(black_frost, hada, hombre, caos).

personaje(nahobino, dios, hombre, ley).
personaje(amaterasu, dios, mujer, ley).
personaje(osiris, dios, hombre, neutral).
personaje(loki, dios, hombre, caos).
```

Primero se declaran los hechos que en este caso son los personajes con sus nombres, raza, género y categoría

```
jugar :-
    pregunta_Raza(Raza),
    pregunta_Genero(Genero),
    pregunta_Alineacion(Alineacion),
    findall(Nombre, personaje(Nombre, Raza, Genero, Alineacion), Coincidencias),
    mostrar_Coincidencias(Coincidencias),
    preguntar_otra_vez.

mostrar_Coincidencias([]) :-
    format("Me rindo, ¿Es tu personaje alguien más? (y/n) "), read(Respuesta),
    ((Respuesta == y ; Respuesta == n) ->
        (Respuesta == y -> jugar;
            (Respuesta == n -> true)
        );
        mostrar_Coincidencias([])
    ).

mostrar_Coincidencias([Nombre|Resto]) :-
    format("Tu personaje podría ser ~w. ¿Es correcto? (y/n) ", [Nombre]),
    read(Respuesta),
    ((Respuesta == y ; Respuesta == n) ->
        (Respuesta == y -> format("¡Felicidades! Tu personaje es ~w.~n", [Nombre]);
            (Respuesta == n -> mostrar_Coincidencias(Resto))
        );
        mostrar_Coincidencias([Nombre|Resto])
    ).
```

Al empezar el sistema experto este comienza a realizar las preguntas y guarda las respuestas guardando los datos, a continuación, compara si las opciones seleccionadas coinciden con uno de los personajes declarado anteriormente, si se encuentra una coincidencia te dice el nombre del personaje con el que coincide y te pregunta si la respuesta es correcta, si es correcto el sistema gana si no es correcto el jugador gana

```
pregunta_Raza(Raza) :-
    format("¿Tu personaje es humano? (y/n) "), read(Respuesta),
    ((Respuesta == y ; Respuesta == n) ->
        (Respuesta == y -> Raza = humano;
            (Respuesta == n ->
                format("¿Tu personaje es un demonio? (y/n) "), read(Respuesta1),
                ((Respuesta1 == y ; Respuesta1 == n) ->
                    (Respuesta1 == y -> Raza = demonio;
                        (Respuesta1 == n ->
                            format("¿Tu personaje es un hada? (y/n) "), read(Respuesta2),
                            ((Respuesta2 == y ; Respuesta2 == n) ->
                                (Respuesta2 == y -> Raza = hada;
                                    (Respuesta2 == n -> Raza = dios)
                                );
                                pregunta_Raza(Raza)
                            )
                        )
                    );
                    pregunta_Raza(Raza)
                )
            )
        )
    ;
        pregunta_Raza(Raza)
    ).
```

La regla pregunta_Raza, se encarga de preguntar al jugador por las diferentes posibles razas para el personaje, va haciendo las preguntas para luego mostrar una recuadro que permite al jugador escribir Y/N dependiendo si su respuesta es Si o No, si la respuesta es Si asume que la raza del personaje corresponde a la que pregunto, si la respuesta es No continua preguntando hasta que recorre todas las razas posibles, esto es posible utilizando métodos recursivos

```
pregunta_Genero(Genero) :-
    format("¿Es tu personaje hombre? (y/n) "), read(Respuesta),
    ((Respuesta == y ; Respuesta == n) ->
        (Respuesta == y -> Genero = hombre;
            (Respuesta == n -> Genero = mujer)
        );
        pregunta_Genero(Genero)
    ).

pregunta_Alineacion(Alineacion) :-
    format("¿Tu personaje es alineación ley? (y/n) "), read(Respuesta),
    ((Respuesta == y ; Respuesta == n) ->
        (Respuesta == y -> Alineacion = ley;
            (Respuesta == n ->
                format("¿Tu personaje es alineación neutral? (y/n) "), read(Respuesta1),
                ((Respuesta1 == y ; Respuesta1 == n) ->
                    (Respuesta1 == y -> Alineacion = neutral;
                        (Respuesta1 == n -> Alineacion = caos)
                    );
                    pregunta_Alineacion(Alineacion)
                )
            )
        )
    ;
        pregunta_Alineacion(Alineacion)
    ).
```

El mismo sistema se aplica para el resto de preguntas hasta que el sistema experto termina de realizar las preguntas

```
preguntar_otra_vez :-
    write("¿Quieres volver a jugar? (y/n) "),
    read(Respuesta),
    ((Respuesta == 'y' ; Respuesta == 'n') ->
        (Respuesta == 'y' -> jugar;
            (Respuesta == 'n' ->
                write("Gracias por jugar conmigo"), nl
            )
        );
        preguntar_otra_vez
    ).
```

La regla preguntar_otra_vez pregunta al jugador si quiere volver a jugar si la respuesta es positiva vuelve a iniciar el juego

```
¿Tu personaje es humano? (y/n)
n
¿Tu personaje es un demonio? (y/n)
y
¿Es tu personaje hombre? (y/n)
y
¿Tu personaje es alineación ley? (y/n)
n
¿Tu personaje es alineación neutral? (y/n)
n
Tu personaje podría ser demi_fiend. ¿Es correcto? (y/n)
n
Me rindo, ¿Es tu personaje alguien más? (y/n)
y
¿Tu personaje es humano? (y/n)
y
¿Es tu personaje hombre? (y/n)
n
¿Tu personaje es alineación ley? (y/n)
y
Tu personaje podría ser nozomi. ¿Es correcto? (y/n)
y
¡Felicidades! Tu personaje es nozomi.
¿Quieres volver a jugar? (y/n)
n
Gracias por jugar conmigo
```

En este ejemplo se realizaron 2 preguntas al sistema experto y este contesto según el personaje que coincidía con las respuestas dadas por el jugador

Ejemplo Enfoque Conexionista (Juego Piedra, Papel y Tijeras)

```python
# Historial de jugadas del usuario
historial = defaultdict(int)

opciones = ["piedra", "papel", "tijeras"]

def predecir_jugada():
    if not historial:
        return random.choice(opciones)  # Primera vez es aleatorio
    return max(historial, key=historial.get)  # Predice la jugada más frecuente del usuario
```

Primero creamos historial el cual es un diccionario donde se almacenan las veces que el jugador eligió cada opción y la lista opciones tiene los 3 posibles movimientos.

Se crea el método predecir_jugada y primero pregunta si no se cuenta con historial, en caso de no tener la opción de la computadora será al azar entre las 3 posibles, si ya se cuenta con información en el historial se elegirá la jugada que más ha repetido el jugador

```python
def jugar():
    victorias_ia = 0
    victorias_usuario = 0

    while victorias_ia < 2 and victorias_usuario < 2:
        eleccion_usuario = input("Elige piedra, papel o tijeras: ").lower()

        if eleccion_usuario not in opciones:
            print("Elección inválida, intenta de nuevo.")
            continue

        eleccion_ia = predecir_jugada()
        print(f"IA eligió: {eleccion_ia}")

        # Almacenar la elección del usuario para futuras predicciones
        historial[eleccion_usuario] += 1
```

Se crea el método jugar junto con las variables del número de victorias para el jugador y para la IA, el juego sucede dentro de un while el cual seguirá hasta que alguno de los 2 gane 2 veces.

El juego inicia preguntándole al jugador que movimiento va a realizar, el jugador debe de escribir una de las 3, si escribe una opción incorrecta se le notificara que no es válida.

A continuación, la IA realizara su movimiento basándose en el historial para intentar predecir al jugador o si no de forma aleatoria, se le notifica al jugador que selecciona la IA y se guarda la opción que el eligió.

```
    if eleccion_usuario == eleccion_ia:
        print("¡Empate!")
    elif (eleccion_usuario == "piedra" and eleccion_ia == "tijeras") or \
            (eleccion_usuario == "papel" and eleccion_ia == "piedra") or \
            (eleccion_usuario == "tijeras" and eleccion_ia == "papel"):
        print("¡Ganaste esta ronda!")
        victorias_usuario += 1
    else:
        print("¡La IA gana esta ronda!")
        victorias_ia += 1

    print(f"Marcador: Usuario {victorias_usuario} - {victorias_ia} IA")

if victorias_usuario == 2:
    print("¡Felicidades! Ganaste.")
else:
    print("La IA ha ganado.")
```

Se comprueba cuáles fueron las elecciones de ambos, si fue la misma se considera un empate, si el jugador selecciono la opción ganadora ganara una ronda, de lo contrario la IA ganara la ronda y se sumara al contador de victorias, mostrando en pantalla cuantas victorias se llevan, cuando alguno de los 2 llega a las 2 victorias termina el ciclo y se imprime cual fue el ganador.

```
Elige piedra, papel o tijeras: piedra
IA eligió: tijeras
¡Ganaste esta ronda!
Marcador: Usuario 1 - 0 IA
Elige piedra, papel o tijeras: tijeras
IA eligió: piedra
¡La IA gana esta ronda!
Marcador: Usuario 1 - 1 IA
Elige piedra, papel o tijeras: piedra
IA eligió: piedra
¡Empate!
Marcador: Usuario 1 - 1 IA
Elige piedra, papel o tijeras: papel
IA eligió: piedra
¡Ganaste esta ronda!
Marcador: Usuario 2 - 1 IA
¡Felicidades! Ganaste.
```

En esta partida el jugador gano 2-1 a la IA

Proceso de aprendizaje automático

Adquisición de datos

Se recopilan los datos que serán utilizados para llevar a cabo el entrenamiento del modelo, estos pueden venir de diferentes fuentes como pueden ser:

-Bases de datos

-Sensores

-APIs de terceros

-archivos CSV o Excel

-Redes sociales

-Imágenes

-Texto

Procesamiento de datos

Una vez obtenidos los datos debe de llevarse a cabo una limpieza y transformación de datos para que se encuentren en el formato adecuado

Limpieza de datos:

Eliminar valores nulos o duplicados

Corregir datos incorrectos

Transformación de datos:

Normalizar valores

Selección de características importantes (Fe ature Selección)

Entrenamiento del modelo

Una vez los datos se encuentren preparados se comienza a entrenar el modelo a partir de patrones, esto se divide en dos

Entrenamiento:

El modelo ajusta sus parámetros con los datos de entrenamiento

Validación:

Se evalúa el modelo en un conjunto de datos no visto antes para ajustar hiperparametros

Para llevar a cabo esto primero se selecciona un algoritmo de machine Learning, puede ser regresión, red neuronal, árbol de decisión, se entrena el modelo con datos etiquetados usando aprendizaje supervisado y con datos no etiquetados usando aprendizaje no supervisado, el modelo ajusta sus parámetros iterativamente para minimizar errores y se continúa entrenando

Evaluación del modelo

Posterior al entrenamiento se mide cual fue el desempeño del modelo con los datos de prueba (testing set) para asegurarse de que generaliza bien.

Dependiendo de cuál sea el problema se utilizan diferentes métricas de evaluación algunos ejemplos son los siguientes:

Precisión: Según el porcentaje de predicciones correctas

Matriz de confusión: evalúa falsos positivos y falsos negativos

F1-score: balance entre precisión y recall

Error cuadrático medio: Diferencia entre valores reales y los predichos

Coeficiencia de determinación: que tan bien el modelo explica la variabilidad de los datos

Similitudes entre Modelo Cognitivo y Aprendizaje automático

Entrada de información:

Ambos inician con una fase de entrada de datos

Procesamiento de información:

Se filtra y organiza la información antes de tomar decisiones

Aprendizaje y adaptación:

Se basan en la detección de patrones para mejorar la toma de decisiones

Toma de Decisiones:

Se produce una salida basada en la información previa

Evaluación y retroalimentación:

Ambos evalúan el rendimiento y ajustan el funcionamiento

Diferencias entre Modelo Cognitivo y Aprendizaje automático

Naturaleza del aprendizaje:

Uno es natural y biológico mientras que el otro es artificial y digital

Flexibilidad:

La mente humana tiene mayor capacidad de adaptación

Memoria y almacenamiento:

El cerebro tiene memoria asociativa mientras que la IA usa datos estructurados

Capacidad de generalización:

Los humanos aprenden con menos datos que un algoritmo

Toma de decisiones éticas y contextuales:

La IA no tiene juicio moral o emociones