Democracia Directa a Posteriori: Un sistema de toma de decisiones grupales basado en Inteligencia Artificial

Miguel Tasende - Sebastián Laborde - Juan Pablo Montesano

Motivación

Grupo que toma decisiones periódicamente

Las aspiraciones y requerimientos de cada individuo (“objetivos individuales”) del grupo tienen el mismo peso, pero…

las opiniones sobre cómo alcanzar esos objetivos (“decisiones ejecutivas individuales”) son muy variadas (y a veces se alejan mucho de los mismos).

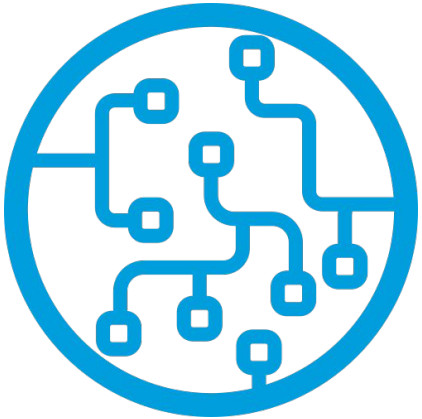
¿cómo combinar esas “decisiones ejecutivas” para lograr satisfacer lo mejor posible los objetivos individuales?

Propuesta

Sistema IA aprende a combinar las decisiones individuales

Dada una combinación de decisiones individuales, produce una decisión grupal

Tiempo después se evalúa la decisión tomada (por un sistema de votación clásico), y el sistema se actualiza para la próxima decisión

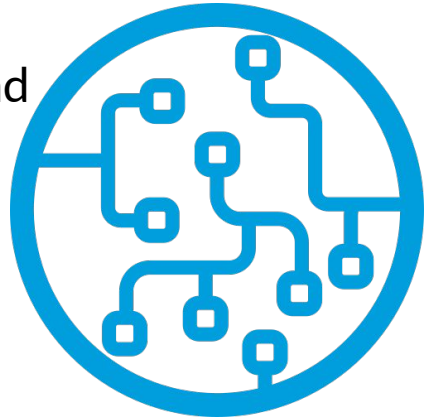


Beneficios

Ejecutividad: Las decisiones se toman de manera inmediata

Criterio Democrático (“a posteriori”): Las decisiones tomadas, si el algoritmo es eficiente en sus predicciones, serán las que logren el mejor consenso entre los integrantes (el criterio específico se explica más adelante).

Decisión Racional: Dado el objetivo planteado, la racionalidad de la decisión depende sólo de la eficiencia del algoritmo en alcanzar el objetivo, y no de emociones y sesgos momentáneos.

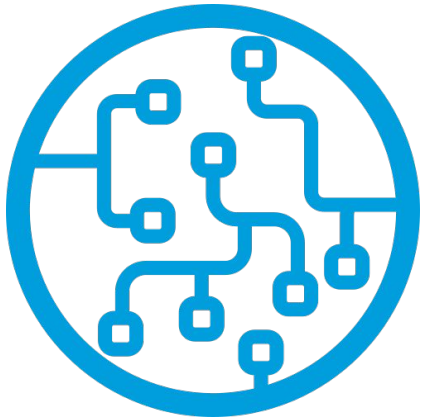


Agregados opcionales

Decisores artificiales: Se pueden unir de forma natural al sistema, sólo en la etapa de decisión (la evaluación la realizan sólo humanos)

Parámetros que describan el problema: ayuda en los casos en que los problemas no son siempre “del mismo tipo”

Para la evaluación se puede usar “Voto Mayoritario”, pero también otros sistemas que produzcan decisiones “de compromiso”



Caso de uso I

Oficina que tiene que procesar datos.

Hay tres opciones: “Automatizar”, “Procesar a mano”, “Subcontratar el servicio”

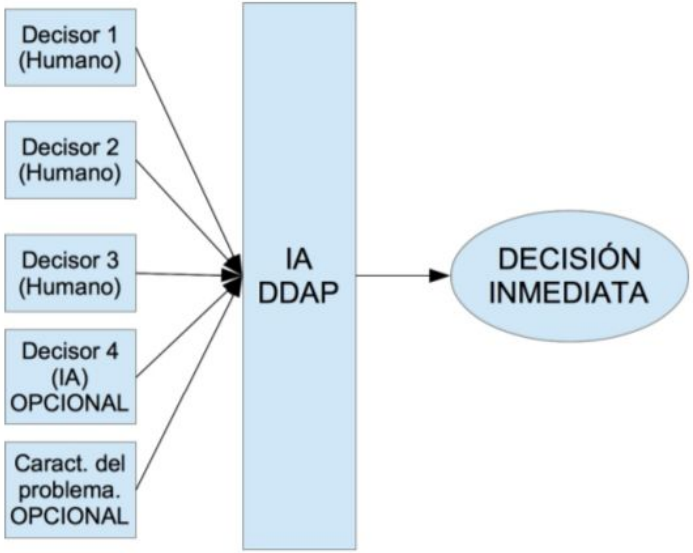
Los funcionarios tienen distintas habilidades y distinto nivel de trabajo en cada instante, y deben decidir “qué hacer” cada vez.

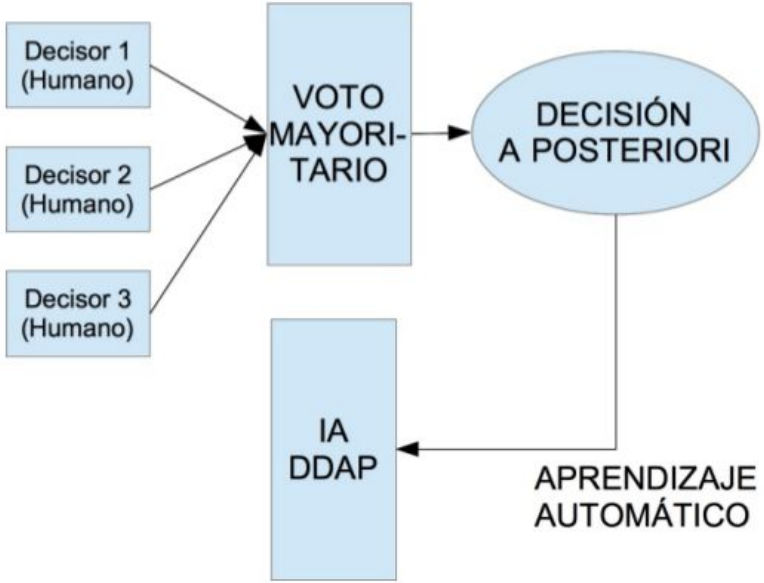
Caso de uso II

Decisiones de Centros Comunales Zonales

Decisiones de pequeños pueblos o ciudades

Diseño


Decisor 1 (Humano)
Decisor 2 (Humano)
Decisor 3 (Humano)
Decisor 4 (IA) OPCIONAL
Caract. del problema. OPCIONAL
IA DDAP
DECISIÓN INMEDIATA


Decisor 1 (Humano)
Decisor 2 (Humano)
Decisor 3 (Humano)
VOTO MAYORI-TARIO
DECISIÓN A POSTERIORI
IA DDAP
APRENDIZAJE AUTOMÁTICO

Ética en el diseño del sistema

La etapa de elección del “sistema de evaluación” es crítica desde el punto de vista de la ética. Se pueden usar distintos sistemas de votación clásicos:

p. ej.: Voto Mayoritario Simple,

Voto por el Método Condorcet: Busca una decisión de compromiso,

etc…

En la etapa de “decisión ejecutiva” y “aprendizaje automático” se busca lograr la mejor concordancia con el objetivo propuesto. Se busca eficiencia.

Si se agregan “parámetros del problema”, hay que cuidar que no introduzcan sesgos y en ese caso el problema es bastante más complicado.

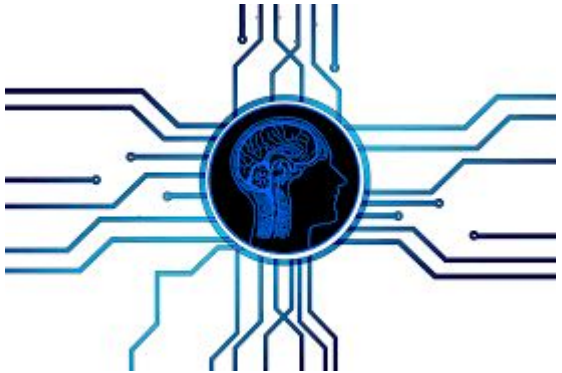
Implementación

Aprendizaje Automático para entrenar la IA

Regresión Logística (Sólo para pruebas en prototipos)

Gradient Boosting (XGBoost, LightGBoost)

Redes Neuronales



Prueba de concepto

Juego **Knights Isolation**

Tablero de ajedrez, dos jugadores se mueven como el caballo

Cada casilla visitada es bloqueada

El objetivo es dejar sin movimientos al oponente

Jugador1: Grupo de jugadores con diferentes algoritmos:

“Random”, “Greedy”, “AlphaBeta+It. Deepening”

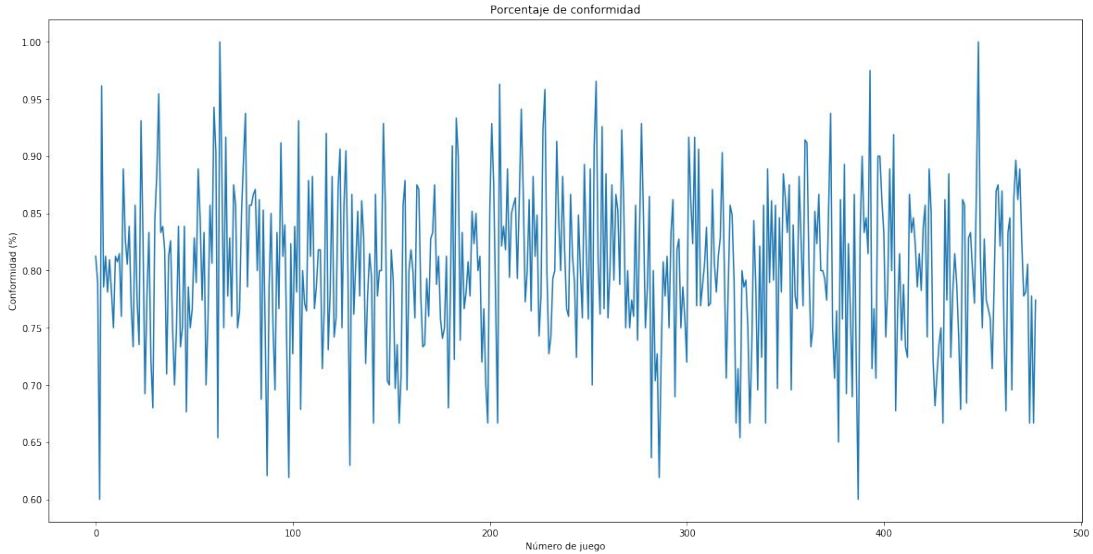
Sistema DDAP para decidir la jugada

Jugador2: Jugador con algoritmo MiniMax

Imagen tomada de Udacity

Resultados iniciales

Se jugaron 500 partidas “virtuales” con Logistic Regression y mirando 2 jugadas adelante para la evaluación.




```python
from sklearn.linear_model import LogisticRegression
from sklearn.multiclass import OneVsRestClassifier
from scipy import stats
import copy
import isolation
import numpy as np


class DDAP(object):
    """
    A class that records all the action-values that the players considered for
    each round in the game.
    """

    def __init__(self,
                 player_names=None,
                 player_time=0.5,
                 predictor=None,
                 look_ahead=2):
        self.player_names = player_names
        self.player_time = player_time
        if predictor is None:
            self.predictor = OneVsRestClassifier(LogisticRegression())
        else:
            self.predictor = predictor
        self.d_values = {p_name: dict() for p_name in player_names}
        self.actions = {p_name: dict() for p_name in player_names}
        self.decisions = dict()
        self.encoder = self.get_encoder()
        self.look_ahead = look_ahead
        self.fitted = False
        self.X = None
        self.y = None

    def get_encoder(self):
        actions = [a.name for a in isolation.isolation._ACTIONSET]
        a = np.array([actions]).T
```

Gracias!