Metaheurísticas para el problema de Aprendizaje de Pesos en Características

Stefani Castellanos Erick Silva

Abstract

Inserte una descripción breve del paper.

Introducción

Debido a la gran cantidad de información manejada actualmente, ha surgido la necesidad de reducir el tamaño de dichos datos, sin embargo, esto genera el problema de escoger correctamente que información es relevante para el clasificador.[1]

Reducir el tamaño de los datos es posible a través de los siguientes métodos:

- Seleccionando características del conjunto de datos, lo que reduce el tamaño de las columnas.
- Eliminando las instancias del conjunto de datos que aporte poca información.
- Determinando la importancia de las características, ya que proporciona información que permite guiar al clasificador y así, reducir el tiempo de procesamiento.

Este trabajo se enfocará en esta última, denominado el problema de *Aprendizaje de Pesos en Características*. Se utilizará un algoritmo *greedy* (ávido) denomindado RELIEF, dos metaheurísticas de trayectoría y dos poblacionales, para finalmente compararlas y determinar cual de ellos es más adecuado para el problema.

1 Descripción del problema

Antes de describir el problema APC, es necesario comprender en qué consiste un problema de clasificación; se dispone de un conjunto de posibles clases (C) y un conjunto de datos (X), en donde una instancia x_i es un vector previamente clasificado y de la forma:

$$x_i = \langle f_1, f_2, ..., f_n, c \rangle$$

en donde:

- f_i : el valor de cada característica.
- n : la cantidad de característcas.
- c : la clase a la que pertenece dicha instancia, con $c \in C$.

Es común particionar X en dos subconjuntos que representen el conjunto de entrenamiento y el de prueba, Xe y Xp respectivamente, de manera que Xe pueda ser utilizado para que el algoritmo aprenda los parámetros que le permitan clasificar correctamente todas sus instancias y utilizar Xp para validar los resultados obtenidos.

El problema del APC consiste en optimizar el rendimiento de un clasificador a partir de la inclusión de pesos asociados a las características del problema que ponderan la relevancia de cada una de ellas en el problema de aprendizaje y modifican su valor en el momento de calcular las distancias entre ejemplos. En este caso, el clasificador considerado será el *K-Nearest Neighbor* (K-vecinos más cercanos) con $K = 1$ (1-NN).

El algoritmo K-NN asume que todas las instancias corresponden a puntos en un espacio n-dimensional ($\Re^n$), en donde n es la cantidad de características del conjunto de datos. El vecino más cercano de una instancia está definido en términos de la distancia Euclideana y el algoritmo determina que los k vecinos cercanos pertenecen a la clase más común de ellos. [2]

1.1 Función objetivo

Inserte Función objetivo

$$Max\ tasa\_clas(1\text{-}NN(s)) = 100 \times \frac{aciertos}{total} \qquad (1)$$

sujeto a:

$$w_i = [0, 1] \quad 1 \leq i \leq n$$

donde:

- $W = \langle w_1, ..., w_n \rangle$ es una solución al problema que consiste en un vector de números reales de tamaño n que define el peso que pondera a cada una de las características f.
- 1-NN es el clasificador k-NN con $k = 1$ vecinos generado a partir del conjunto de datos inicial, utilizando la técnica de validación leave-one-out y los pesos en W que se asocian a las n características.

- T es el conjunto de datos sobre el que se evalúa el clasificador, ya sea el conjunto de entrenamiento como el de prueba.
- *aciertos* = nº de instancias bien clasificadas de T
- *total* = nº de total de instancias de T

1.2 Representación

Inserte representación

2 Conjunto de datos

Inserte descripción del conjunto de datos a utilizar

3 Clasificador

4 Algortimo RELIEF

5 Metaheurísticas

5.1 Metaheurísticas de trayectoria

5.1.1 Enfriamiento Simulado (Simulated annealing)

5.1.2 Búsqueda Local Iterada (Iterated Local Search)

5.2 Metaheurísticas poblacional

6 Implementación

Inserte Implementación

7 Resultados

Inserte Resultados

Particion	Aciertos(

Particion	Aciertos()	Error()	Tiempo(sg)
1	46.988	53.012	0
2	54.2169	45.7831	0
3	62.6506	37.3494	0
4	48.1928	51.8072	0

Particion	Aciertos()	Error()	Tiempo(sg)
1	64.1304	35.8696	7
2	63.6413	36.3587	6
3	60.4348	39.5652	6
4	71.087	28.913	7

Particion	Aciertos()	Error()	Tiempo(sg)
1	77.9736	22.0264	0
2	46.2555	53.7445	0
3	41.4097	58.5903	0
4	77.9736	22.0264	0

Conclusiones

Aquí concluyen.

References

[1] Cano, J. Herrera, F. Lozano. M Using evolutionary algorithms as Instance Selection for data reduction in KDD: an experimental study. *IEEE Transaction on Evolutionary computation*, 2003.

[2] Mitchell, T. Machine Learning. *From Book News, Inc*, 1997.

[3] EJEMPLO C. So and H. So. A groundbreaking result. *Journal of Everything*, 59(2):23–37, 2005.

Apéndice

Bla.

8 La sección de ejemplo de la profe

Citan así [3]. ELIMINAR AL ESCRIBIR EL INFORME

Algorithm 1: Nombre

Input: Descripcion
Output: Descripcion

```
1 Primer paso
2 Segundo
3 foreach i = 1...n do
4     if Alguna condición then
5         Algo aqui
6 return Valor
```