

Metaheurísticas para el problema de Aprendizaje de Pesos en Características

Stefani Castellanos

Erick Silva

Abstract

Inserte una descripción breve del paper.

Palabras claves: Aprendizaje de Pesos en Características, Metaheurísticas, *Machine Learning*.

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) denominado RELIEF, dos metaheurísticas de trayectoria 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:

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

en donde:

- f_i : el valor de cada característica.

- n : la cantidad de características.
- 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, X_e y X_p respectivamente, de manera que X_e pueda ser utilizado para que el algoritmo aprenda los parámetros que le permitan clasificar correctamente todas sus instancias y utilizar X_p 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 conjunto de datos. Estos pesos ponderan la relevancia de cada característica en y modifican su valor al momento de calcular las distancias entre instancias, de tal forma que los clasificadores que se construyan a partir de estos pesos sean certeros y/o más rápidos. En este caso en particular, 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 (\mathbb{R}^n), en donde n es la cantidad de características del conjunto de datos. [3]. El algoritmo es el siguiente:

Algorithm 1: K-Nearest Neighbor

Input: Conjunto de entrenamiento X_e , Conjunto de prueba X_p , K

Output: Conjunto X_p clasificados según C

```

1 foreach  $x_i \in X_p$  do
2    $vecinos = []$ ;
3   foreach  $x_j \in X_e$  do
4      $d = distancia(x_i, x_j)$ ;
5      $agregar(x_j, d, vecinos)$ ;
6    $k\_vecinos = seleccionar\_cercanos(k, vecinos)$ ;
7    $clasificar(x_i, k\_vecinos)$ ;
8 return  $X_p$  clasificado

```

El proceso de aprendizaje de este clasificador consiste en almacenar una tabla con las instancias correspondientes al X_e junto a la clase asociada a cada uno de ellos. Dado una instancia $x_i \in X_p$, se

calcula su distancia a todas las otras que pertenecen al conjunto de entrenamiento y se escogen las k más cercanas [2]; usualmente se determina la proximidad entre dos ejemplos utilizando la distancia Euclideana. Finalmente, x_i se clasifica según la clase mayoritaria grupo y se retorna el conjunto de pruebas clasificado.

1.1 Función objetivo

Al ser APC un problema de optimización, es necesario establecer cual es la función que se desea mejorar. En este caso, dadas las características del problema, resulta evidente que obtener una "buena" solución está fuertemente relacionado con la cantidad de instancias clasificadas correctamente usando 1-NN. El objetivo es encontrar el mejor vector de pesos que permita maximizar la tasa de aciertos del clasificador 1-NN. Más específicamente:

$$\text{Max tasa}(1\text{-NN}(X, W)) = 100 \times \frac{\text{aciertos}(X)}{\text{total}(X)} \quad (1)$$

sujeto a:

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

donde:

- $W = \langle w_1, \dots, w_n \rangle$ es una solución al problema.
- 1-NN es el clasificador k-NN con $k = 1$ vecinos, generado a partir del conjunto de datos inicial e incluyendo el peso de W al momento de calcular las distancias.
- X es el conjunto de datos sobre el que se evalúa el clasificador.
- *aciertos* es la cantidad de instancias de X clasificadas correctamente por 1-NN.
- *total* es la cantidad total de instancias de X .

1.2 Representación de la solución

La solución al problema de Aprendizaje de Pesos en Características viene dado por $W = \langle w_1, \dots, w_n \rangle$, un vector de números reales de tamaño n (cantidad de características) en el que el valor de cada w_i define el peso que pondera a la característica f_i , es decir, representa que tan importante para distinguir un ejemplo de otro.

Cada w_i debe pertenecer al intervalo $[0, 1]$; los valores cercanos a 1 indican que la característica es más importante. En caso de que algún valor quede fuera de este intervalo, se debe normalizar, seleccionando al máximo valor del vector y dividiendo todos los valores entre dicho número.

1.3 Conjunto de datos

Con la expansión del área de Inteligencia Artificial, se ha convertido en una necesidad contar con bases de datos que proporcionen información que permita a los investigadores, educadores y estudiantes del área realizar análisis sobre los algoritmos de *Machine Learning*. Una de las librerías más populares en la actualidad es UCI Machine Learning Repository, que mantiene una colección de conjuntos de datos, teorías de dominio y generadores de datos disponibles para toda la comunidad.

Para efectuar el análisis de las metaherísticas a utilizar para resolver el problema APC se utilizarán cuatro librerías disponibles en UCI:

Iris: este es probablemente el conjunto de datos mejor conocido y citado en la literatura de reconocimiento de patrones. Contiene tres clases de 50 instancias cada una, en donde cada clase corresponde a un tipo de planta Iris. Cada instancia posee las siguientes características: largo del sépalo, ancho del sépalo, largo del pétalo y ancho del pétalo en centímetros. Las clases a predecir son: "Iris-Setosa", "Iris-Versicolor" e "Iris-Virginica". [4].

Sonar: es una base de datos de detección de materiales mediante señales de sónar que "rebotan" en los objetos desde diferentes ángulos y bajo condiciones varias, discriminando entre cilindros metálicos(*mines*) y rocas(*rocks*). Cuenta con 111 *mines* y 97 *rocks* donde cada instancia es un conjunto de 60 números entre 0.0 y 1.0 que representan la energía entre una banda en particular, integrada sobre un periodo dispuestas en orden creciente según el ángulo. Las clases a predecir son: "R" (*rocks*) y "M" (*mines*). [6].

Winsconsin Diagnostic Breast Cancer: es una base de datos contiene atributos calculados a partir de una imagen digitalizada de una aspiración con aguja fina de una masa en la mama. Se describen las características de los núcleos de las células presentes en la imagen. La tarea consiste en determinar si un tumor encontrado es benigno o maligno. Cuenta con 357 tumores benignos y 212 malignos en donde cada instancia posee 30 características, 10 valores reales computados por cada célula: radio, textura, perímetro, área, suavidad, compacto, concavidad, puntos cóncavos, simetría y dimensión fractal. [7].

Spam Base: es una base de datos de detección de SPAM (correo basura) frente a correo electrónico seguro. Cuenta de 460 ejemplos y 57 atributos en

donde los primeros 48 corresponden al porcentaje de frecuencia de una palabra en particular en un correo, 6 corresponden al porcentaje de frecuencia de símbolos de puntuación y las últimas 3 son el promedio, máximo y la suma de la cantidad de letras en mayúsculas. Las clases a predecir son: 1 (spam), 0 (non-spam) que representan [5].

2 Algoritmo RELIEF

Algorithm 2: RELIEF

Input: Conjunto de entrenamiento X_e

Output: Vector de pesos W

```

1  $W = 0, 0 \dots, 0$ 
2 foreach  $x_i \in X_e$  do
3    $a = \text{amigo\_mas\_cercano}(x_i);$ 
4    $e = \text{enemigo\_mas\_cercano}(x_i);$ 
5
6   /* Actualizar pesos de  $W$  */
7   foreach  $w_i \in W$  do
8      $\text{dif\_amigo} = \text{diferencia}(w_i, a);$ 
9      $\text{dif\_enemigo} = \text{diferencia}(w_i, e);$ 
10     $w_i = w_i - \text{dif\_amigo} + \text{dif\_enemigo};$ 
11  $\text{normalizar}(W);$ 
12 return  $W$ 
```

3 Metaheurísticas

3.1 Metaheurísticas de trayectoria

3.1.1 Enfriamiento Simulado (Simulated annealing)

3.1.2 Búsqueda Local Iterada (Iterated Local Search)

3.2 Metaheurísticas poblacional

3.2.1 Algoritmo Genético

3.2.2 Algoritmo Memético

4 Implementación

Inserte Implementación

5 Resultados

Inserte Resultados

Partición	Aciertos(%)	Error(%)	Tiempo(sg)
1	90	10	0
2	91.6667	8.33333	0
3	95	5	0
4	75	25	0

Partición	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

Partición	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

Partición	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] Fisher, R.A. (1988). UCI Machine Learning Repository [<https://archive.ics.uci.edu/ml/machine-learning-databases/iris/>]. Irvine, CA: University of California, School of Information and Computer Science.
- [3] Herrera, F. Metaheurísticas. *Seminario 2: Problemas de optimización con técnicas basadas en búsqueda local*, 2017. Disponible en: <http://sci2s.ugr.es/sites/default/files/files/Teaching/GraduatesCourses/Metaheuristics/Sem02-Problemas-BusquedaLocal-MHs-16-17.pdf>
- [4] Hopkins, M. (1999). UCI Machine Learning Repository [<https://archive.ics.uci.edu/ml/machine-learning-databases/spambase/>]. Irvine, CA: University of California, School of Information and Computer Science.
- [5] Mitchell, T. Machine Learning. *From Book News, Inc*, 1997.
- [6] Sejnowski, T (año). UCI Machine Learning Repository [<http://archive.ics.uci.edu/ml/machine-learning-databases/undocumented/connectionist-bench/sonar/>]. Irvine, CA: University of California, School of Information and Computer Science.
- [7] Talbi E. Metaheuristics: From desing to implementation *University of Lille*, 2009.
- [8] Wolberg, W. (1995). UCI Machine Learning Repository [<https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/>]. Irvine, CA: University

of California, School of Information and Computer
Science.

Apéndice
Bla.