CafsPy: Una biblioteca de Python de selección de características de matriz de cobertura

Abstracto

CafsPy es una biblioteca de Python de selección de características basada en la teoría de arreglos de cobertura (CA). Permite el descubrimiento de subconjuntos mínimos pero altamente informativos de características o longitudes de onda, optimizando la precisión de la clasificación en conjuntos de datos de alta dimensión. CafsPy es particularmente valioso en dominios como la quimiometría, donde cientos o miles de bandas espectrales requieren reducción antes de la clasificación.

> *"CafsPy se desarrolló como un módulo de selección de características basado en Python que genera múltiples subconjuntos de características de alta precisión. Proporciona a los profesionales e investigadores la capacidad de seleccionar subconjuntos que mantienen una F1-score adecuada".*
>
> (Romo et al., 2025, p. 3)

Visión general

CafsPy es una biblioteca basada en Python diseñada para la selección de características y bandas de onda mediante la aplicación de **arreglos de cobertura (CA, por sus siglas en inglés).** Aborda específicamente el desafío de los datos de alta dimensión mediante el modelado de interacciones entre características utilizando un enfoque combinatorio estructurado. La biblioteca implementa dos algoritmos principales: **Covering Array Feature Selection (CAFS)** y **Iterative Covering Array Feature Selection (ICAFS),** ambos basados en el concepto de pruebas de *interacción de t-way comúnmente utilizadas en ingeniería de software*.

Los arreglos de cobertura se representan como **CA(N, k, t, v),** donde:

- **N** denota el número de casos de prueba (filas),
- **k** es el número de parámetros (características),
- **t** representa la fuerza de la interacción,
- **v** es el tamaño del alfabeto (típicamente binario para CafsPy).

Cada fila de la CA corresponde a un subconjunto de características que luego se evalúa mediante un modelo de clasificación. El **algoritmo CAFS** utiliza una CA fija para reducir iterativamente el conjunto de características, mientras que **ICAFS** regenera una nueva CA en cada iteración en función del conjunto de características actualizado utilizando el **algoritmo IPOG**, preservando así la cobertura de interacción.

La implementación de ICAFS es altamente configurable: los usuarios pueden seleccionar el clasificador (por ejemplo, kNN, SVM, MLP), definir la fuerza de interacción *t*, establecer el número de iteraciones *T* y aplicar la mezcla basada en semillas para la generación de subconjuntos aleatorios pero reproducibles. A diferencia de las técnicas tradicionales de selección de características, CafsPy enfatiza el descubrimiento de **subconjuntos que mantienen o mejoran el rendimiento de la clasificación** (medido a través de la F1-score), no simplemente identificando las características más relevantes individualmente.

Algoritmos implementados

La biblioteca incluye dos algoritmos principales:

1. CAFS (Covering Array Feature Selection): utiliza un arreglo de cobertura binario estático para generar varios subconjuntos de características y evalúa el rendimiento de la clasificación en cada uno de ellos. Está limitado al tamaño del arreglo de cobertura inicial (1401 características como máximo). Además, asume dependencias de características fijas y utiliza un arreglo truncado como características a reducir.
2. ICAFS (Iterative CAFS): Es una mejora con respecto a CAFS, dado que genera un nuevo arreglo de cobertura en cada iteración utilizando el algoritmo IPOG. La ventaja es que mantiene las interacciones de la vía t dinámicamente a medida que se reducen las características, mejorando la adaptabilidad. Además, funciona con cualquier clasificador sklearn, como kNN, SVM, MLP, OPF, etc., lo que proporciona independencia del clasificador.

Cómo funciona

Ejemplo de uso en Python:

```
from cafspy import ICAFS, CAFS
from sklearn.neighbors import KNeighborsClassifier

# Preparación de datos
X = (data - data.min()) / (data.max() - data.min())
y = labels

# Establece los hiperparámetros y el clasificador
clf = KNeighborsClassifier(n_neighbors=3)

# Ejecuta ICAFS
scores_list, feature_list = ICAFS(X, y, t=2, T=10, lr=clf, print_logs=True)
```

Aplicaciones

CafsPy ha demostrado un sólido rendimiento en quimiometría, imágenes hiperespectrales y tareas de clasificación general. Por ejemplo, en la clasificación de **nibs de clones de cacao amazónicos**, CAFS pudo reducir 1401 longitudes de onda NIR a solo cinco, logrando una puntuación F1 del 98,89%, superando a otras técnicas como la selección de características de centralidad de vectores propios (ECFS) y la selección de características de múltiples grupos (MCFS) (Castro et al., 2022).

Taxonomía de selección de características

La selección de características (FS) es un paso previo al procesamiento vital en el aprendizaje automático y el análisis de datos, con el objetivo de reducir la dimensionalidad de los datos al tiempo que se preserva o mejora la precisión e interpretabilidad del modelo. Los métodos tradicionales de FS se dividen en **tres grandes categorías**:

1. **Métodos de filtro.** Evaluar la importancia de las características basadas en medidas estadísticas (p. ej., correlación, información mutua) independientemente de cualquier algoritmo de aprendizaje.
2. **Métodos de envoltura.** Utilice el rendimiento de un modelo de aprendizaje específico para evaluar diferentes subconjuntos de características, que a menudo implican estrategias de búsqueda como la selección hacia adelante, la eliminación hacia atrás o las metaheurísticas.
3. **Métodos incrustados.** Realice la selección de características como parte del proceso de entrenamiento del modelo.

CafsPy es un **método basado en envoltorios** bajo la categoría de **selección de subconjuntos**, que utiliza específicamente **estrategias de búsqueda combinatoria (arreglos de cobertura)** para explorar el espacio de características. Cada subconjunto se evalúa mediante un modelo de aprendizaje automático con validación cruzada y se selecciona el subconjunto con mejor rendimiento. Esto convierte a CafsPy en un **selector de subconjuntos específico del modelo**, como se define en taxonomías de FS más recientes, como las propuestas por **Li et al. (2017).**

A diferencia **de los envoltorios metaheurísticos** (por ejemplo, algoritmos genéticos, algoritmos binarios de murciélagos), que pueden carecer de garantías de reproducibilidad o cobertura teórica, CafsPy proporciona **una generación de subconjuntos sistemática y teóricamente fundamentada** a través de CA. Además, llena un vacío en el ecosistema de Python, que carece en gran medida de herramientas basadas en la selección de características impulsadas por CA a pesar de su éxito en las pruebas de software y el diseño de sistemas.

Instalación y requisitos

Instalar a través de pip:

pip install cafspy

Referencias

Castro, W., De-la-Torre, M., Avila-George, H., Torres-Jimenez, J., Guivin, A., & Acevedo-Juárez, B. (2022). Amazonian cacao-clone nibs discrimination using NIR spectroscopy coupled to naïve Bayes classifier and a new waveband selection approach. Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy, 270, 120815. https://doi.org/10.1016/j.saa.2021.120815

Castro, W., Seminario, R., Nauray, W., Acevedo-Juárez, B., De-la-Torre, M., & Avila-George, H. (2025). Multispectral drone imagery dataset for plus and non-plus Neltuma pallida trees in northern Peru. Data in Brief, 60, 111645. https://doi.org/10.1016/j.dib.2025.111645

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., & Liu, H. (2017). Feature selection: A data perspective. *ACM Computing Surveys (CSUR)*, *50*(6), 1–45. https://doi.org/10.1145/3136625

Romo Macías, S., Avila-George, H., Torres-Jimenez, J., Castro, W., & De-la-Torre, M. (2025). CafsPy: A Covering Array Feature Selection Python Library. In Proceedings of the International Conference on Software Process Improvement (en revisión).