Entrenamiento de una Máquina de Soporte Vectorial usando Algoritmos Evolutivos con Manejador de Restricciones

TC3023: Inteligencia Computacional

Fecha límite de entrega: 12 de septiembre de 2019

1 Idea general

La idea es entrenar una Máquina de Soporte Vectorial (SVM, por sus siglas en inglés) usando un Algortimo Evolution (EA, por sus siglas en inglés) adaptando un manejador de restricciones. El problema de aprendizaje de una SVM es formulado como un problema de optimización con restricciones. Matemáticamente, el problema de optimización es definido como:

$$\begin{aligned} \min & \frac{1}{2}\sum_{i,j=1}^{m}\alpha_i\alpha_j y_i y_j K\left(\mathbf{x}_i,\mathbf{x}_j\right)-\sum_{i=1}^{m}\alpha_i \\ \text{sujeto a: } & \sum_{i=1}^{m}\alpha_i y_i = 0 \\ & 0\leq\alpha_i\leq C, \\ & \forall i\in\{1,\ldots,m\} \end{aligned} \tag{1}$$

donde $\mathbf{x}_1$ y y_i es la i-ésima instancia de entrenamiento y su respectiva etiqueta de clase, α_i es un multiplicador de Lagrange asociado a cada instancia de entrenamiento y C es el factor de penalización y $K\left(\mathbf{x}_i,\mathbf{x}_j\right)$ es una función kernel.

2 Entrenamiento

La SVM recibe como entrada una matriz $\mathbf{X}\in\mathbb{R}^{m\times n}$, que es el conjunto de entrenamiento con m ejemplos y n características, y un vector $\mathbf{Y}\in\{-1,1\}^{m\times 1}$, que indica la etiqueta de clase de cada ejemplo.

El entrenamiento de la SVM involucra resolver el problema de optimización descrito en la Ecuación (1). Maximizar la función es equivalente a encontrar la instancias que definen el hiperplano de máxima separación. A estas instancias se les conocerá como Vectores de Soporte. Los vectores de soporte corresponden a todas las soluciones donde $\alpha_i > 0$, mientras que las instancias donde $\alpha_i = 0$ no son vectores de soporte.

Para resolver el problema de optimización, se necesita computar una función kernel, descritas en la sección 2.1.

2.1 Funciones Kernel

Algunas funciones kernel a usar son las siguientes:

- Lineal:
$$K\left(\mathbf{x}_i, \mathbf{x}_j\right) = \mathbf{x}_i \cdot \mathbf{x}_j^T \tag{2}$$

- Polinomial:
$$K\left(\mathbf{x}_i, \mathbf{x}_j\right) = \left(\mathbf{x}_i \cdot \mathbf{x}_j^T + C_0\right)^d \tag{3}$$
donde d es el exponente del polinomial y C_0 es una constante.

- RBF:
$$K\left(\mathbf{x}_i, \mathbf{x}_j\right) = e^{-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2} \tag{4}$$
donde γ es un parámetro del kernel.

2.2 Criterios de Paro

Algunos posibles criterios de paro que pueden utilizar son los siguientes:

- Máximo número de evaluaciones de la función objetivo.
- Convergencia, estimado en que no haya habido una mejora en la red durante un determinado número de generaciones; por ejemplo, que durante 10 generaciones no se ha encontrado una red con un error menor.

2.3 Restricciones

Se debe considerar la implementación de al menos dos manejadores de restricciones, a elegir entre:

- Separación de objetivos y restricciones mediante co-evolución.
- Reglas de factibilidad.
- ϵ-constraint.
- Rankeo estocástico.

3 Predicción

Una vez que la SVM esté entrenada, el resultado de la optimización, son los valores vectores de soporte. Estos valores son usados para predecir los valores de clase de nuevas instancias. El valor de salida de una nueva instancia $\mathbf{x}$ es calculado como:

$$y^* = \text{sign}\left(\sum_{i \in SV} \alpha_i y_i K\left(\mathbf{x}_i, \mathbf{x}\right) + b\right) \tag{5}$$

donde b es determinado como:

$$b = \frac{1}{2}\left(\max\left(\sum_{i \in SV, y=-1} \alpha_i y_i K\left(\mathbf{x}_i, \mathbf{x}\right)\right) + \min\left(\sum_{i \in SV, y=1} \alpha_i y_i K\left(\mathbf{x}_i, \mathbf{x}\right)\right)\right) \tag{6}$$

4 Entregables

El objetivo es usar un EA con un manejador de restricciones para entrenar una SVM. Posteriormente, hacer uso de dichos pesos para predecir (clasificar) casos no vistos. Por tanto, se debe entregar, como mínimo[1], las implementaciones que permitan hacer el entrenamiento y la predicción de nuevos ejemplos.

model = trainEvoSVM(X_training, Y_training, arguments);
yp = model.evalEvoSVM(X_test);

donde model contiene toda la información del modelo aprendido después de la optimización y yp son las salidas predichas por Model.

5 Evaluación

Los criterios de evaluación son los siguientes:

- **Funcionalidad** (0–15 puntos). El código corre sin errores y funciona en todos los casos.
- **Implementación** (0–15 puntos). Cada uno de los algoritmos son implementados correctamente y representan o resuelve adecuadamente el problema planteado.
- **Restricciones** (0–20 puntos). Se ha considerado al menos dos manejadores de restricciones, implementados y adaptados de manera adecuada al algoritmo de optimización.
- **Funciones objetivo** (0–20 puntos). Se ha implementado de manera adecuada todas la función objetivo, así como todas aquellas requeridas para su cómputo.
- **Desempeño** (0–20 puntos). El algoritmo implementado tiene el desempeño esperado; es decir, el desempeño en términos de desempeño es a lo más un 7% inferior al error de base.
- **Documentación** (0–10 puntos). El código es documentado apropiadamente, incluyendo la información del estudiante y la leyenda del código de ética.

Asimismo, se debe añadir la siguiente leyenda:

> "Apegándome al Código de Ética de los Estudiantes del Tecnológico de Monterrey, me comprometo a que mi actuación en este examen esté regida por la honestidad académica."

Nota: Cualquier consideración no prevista en el presente documento, se tendrá que plantear en el aula de clases (o, en su defecto, se abrirá un grupo de discusión) y se definirán los criterios aplicables en tales situaciones.

[1] Pueden implementarse más funciones a fin de modularizar el código.