**Nombre del Algoritmo:**  Cascading classifiers

**Descripción del Problema Computacional para la cual el algoritmo es una solución.**

**Delimitación del Problema:**

El algoritmo de cascading classifiers está diseñado para abordar problemas de clasificación en los que es crucial optimizar tanto la precisión como la eficiencia computacional. Este enfoque es especialmente útil para detectar patrones en grandes volúmenes de datos, como en tareas de reconocimiento facial o detección de objetos, donde la mayoría de las muestras no pertenecen a la clase objetivo.

**Comprensión del Problema:**

El problema principal es identificar instancias positivas (clase objetivo) en un conjunto de datos mediante una cascada de clasificadores que filtran progresivamente instancias negativas, reduciendo así el costo computacional al concentrarse solo en las muestras relevantes.

**Clasificación del Problema:**

* Aprendizaje Supervisado: El algoritmo utiliza clasificadores entrenados.
* Clasificación Binaria: Se clasifica en positivo (clase objetivo) y negativo.
* Jerárquico/Ensamble: Combina múltiples clasificadores en una estructura secuencial.

**Viabilidad Computacional:**

La cascada es computacionalmente viable porque permite descartar rápidamente muestras irrelevantes con clasificadores simples, reservando los clasificadores más complejos para las muestras más difíciles.

**Contexto del Problema:**

Los cascading classifiers son particularmente efectivos en problemas de detección de objetos donde:

* Solo una pequeña proporción de muestras pertenece a la clase objetivo.
* Los recursos computacionales son limitados.

Ejemplos incluyen:

* Reconocimiento facial en imágenes.
* Detección de defectos en sistemas de manufactura.
* Sistemas de seguridad basados en imágenes.

**Análisis de Datos y Estructura del Problema:**

* Entrada: Conjunto de datos etiquetados 𝐷 con características 𝑋 y etiquetas 𝑌.
* Estructuras: Cadena jerárquica de clasificadores 𝐶1, 𝐶2,…,𝐶𝑛, donde cada 𝐶𝑖 filtra instancias y pasa solo las positivas al siguiente clasificador.

**Evaluación de Complejidad:**

* Entrenamiento: Depende del número de clasificadores 𝑛 y la complejidad de cada 𝐶𝑖.
* Predicción: La complejidad promedio depende de cuántas instancias son descartadas en etapas tempranas.

**Análisis del Contexto del Problema para la cual el algoritmo es una solución**.

**Contexto general del Problema:**

Los cascading classifiers se utilizan dónde:

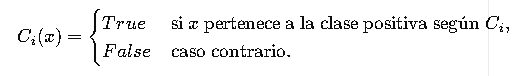
* Es esencial minimizar falsos positivos para evitar errores.
* Las instancias negativas son mucho más comunes que las positivas.
* La eficiencia computacional es crítica, como en sistemas en tiempo real.

**Identificación de los conjuntos y estructuras de datos:**

* Conjunto de datos:
  + Datos etiquetados con 𝑑 características.
  + Representación numérica adecuada para los clasificadores seleccionados.
* Estructura:
  + Cascada jerárquica de clasificadores con parámetros ajustados para maximizar precisión y eficiencia.

**Formalización del problema:**

El objetivo es construir una cascada de clasificadores {𝐶1,𝐶2,…,𝐶𝑛} que minimice los falsos negativos y optimice la eficiencia. Cada clasificador 𝐶𝑖 aplica una función de decisión:



**Análisis de la Dinámica del Problema:**

1. Etapas Iniciales:

* Clasificadores rápidos descartan la mayoría de las muestras negativas.

1. Etapas Intermedias:

* Clasificadores moderadamente complejos procesan las instancias restantes.

1. Etapas Finales:

* Clasificadores avanzados aseguran alta precisión en las muestras más difíciles.

**Evaluación de la Complejidad del Contexto:**

* Ventajas
  + Reduce el costo computacional al filtrar progresivamente.
  + Maneja grandes volúmenes de datos.
* Desventajas:
  + Requiere un diseño cuidadoso de cada clasificador.
  + Errores acumulativos en la cascada pueden afectar el desempeño.

**Identificación de los casos límites y excepciones:**

**Casos límite:**

* Clases altamente desequilibradas.
* Clasificadores iniciales muy débiles que dejan pasar demasiados falsos positivos.

**Excepciones:**

* Datos ruidosos o características irrelevantes pueden degradar el desempeño.

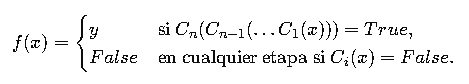
**Interacción con otros sistemas:**

* Puede integrarse en sistemas de procesamiento de imágenes o series temporales.
* Utilizable como filtro previo para otros algoritmos más complejos.

**Especificación Formal:**

**Definición de las entradas y salidas:**

* Entradas:
  + Conjunto de datos 𝐷={(𝑥1,𝑦1),…,(𝑥𝑛,𝑦𝑛)}.
  + Clasificadores 𝐶1,𝐶2,…,𝐶𝑛.
* Salida:
  + Predicción de clase para cada instancia 𝑥.

**Definición formal del problema mediante funciones:**

**Relaciones y Restricciones:**

* Relaciones:
  + Cada clasificador se aplica en secuencia.
* Restricciones:
  + Los clasificadores deben minimizar falsos negativos.

**Invariantes y propiedades:**

* Cada instancia descartada no pertenece a la clase positiva con alta probabilidad.

**Control de Flujo:**

* Inicializar la cascada con 𝐶1.
* Aplicar cada 𝐶𝑖 en secuencia.
* Detener la cascada si 𝐶𝑖(𝑥) =𝐹𝑎𝑙𝑠𝑒

**Pseudocodigo del Algoritmo**

Algoritmo CascadingClassifiers

// Entradas:

// datos\_entrenamiento: Matriz[n][m], n instancias y m atributos.

// etiquetas: Vector[n], clases asociadas a cada instancia.

// cascada: Lista de clasificadores previamente entrenados.

Funcion EntrenarCascada(datos\_entrenamiento, etiquetas)

Definir cascada como ListaVacia

umbral\_fallo <- 0.1 // Tasa de error permitida para cada clasificador.

Mientras datos\_entrenamiento no esté vacío

// Entrenar un clasificador débil

clasificador <- EntrenarClasificadorDebil(datos\_entrenamiento, etiquetas)

// Evaluar el clasificador

fallos <- EvaluarErrores(clasificador, datos\_entrenamiento, etiquetas)

// Ajustar el conjunto de datos para los falsos positivos

datos\_falsos\_positivos <- FiltrarFalsosPositivos(clasificador, datos\_entrenamiento, etiquetas)

// Añadir clasificador a la cascada

AgregarALista(cascada, clasificador)

// Si los errores son menores al umbral, detener

Si fallos <= umbral\_fallo Entonces

Romper

FinSi

// Actualizar el conjunto de datos

datos\_entrenamiento <- datos\_falsos\_positivos

FinMientras

Retornar cascada

FinFuncion

Funcion PredecirCascada(cascada, instancia)

Para cada clasificador en cascada

resultado <- EvaluarClasificador(clasificador, instancia)

Si resultado = "Rechazar" Entonces

Retornar "Clase Negativa"

FinSi

FinPara

Retornar "Clase Positiva"

FinFuncion

// Proceso principal

Escribir "Entrenando clasificadores en cascada..."

datos\_entrenamiento <- CargarDatos()

etiquetas <- CargarEtiquetas()

cascada <- EntrenarCascada(datos\_entrenamiento, etiquetas)

Escribir "Cascada entrenada con éxito. Realizando predicciones..."

instancia <- CargarNuevaInstancia()

resultado <- PredecirCascada(cascada, instancia)

Escribir "Resultado de la predicción: ", resultado

FinAlgoritmo

**Caso de Usos Posibles:**

1. Reconocimiento facial en imágenes:

* Detectar rostros en tiempo real usando clasificadores basados en características como Haar.

1. Detección de defectos en manufactura:

* Filtrar productos defectuosos a partir de imágenes de alta resolución.

1. Sistemas de vigilancia:

* Identificar intrusos en videovigilancia utilizando clasificadores rápidos para reducir el costo computacional.

1. Filtrado de spam:

* Clasificar correos electrónicos como spam o no spam mediante cascadas basadas en reglas simples seguidas de modelos avanzados.