Curso: Inteligencia Artificial

Unidad 2: Aprendizaje automático

Sesión 13: Aprendizaje Basado en Instancias

Docente: Carlos R. P. Tovar

Universidad Tecnológica del Perú

INICIO

¿Tienen dudas o consultas sobre la clase previa?

OBJETIVO
Objetivos de la sesión

Al finalizar la sesión, los alumnos serán capaces de:

- Comprender los fundamentos del aprendizaje basado en instancias y sus características principales como método de aprendizaje "perezoso" (lazy learning).
- Explicar el funcionamiento del algoritmo k-Vecinos Más Cercanos (k-NN) y los factores que afectan su rendimiento (valor de k, métricas de distancia, normalización de datos).
- Implementar un clasificador k-NN utilizando scikit-learn, incluyendo el preprocesamiento necesario de los datos.
- Evaluar el rendimiento del modelo k-NN mediante métricas de precisión, matriz de confusión y validación cruzada.
- Identificar las ventajas y desventajas del aprendizaje basado en instancias en comparación con otros métodos de aprendizaje automático.
- Aplicar el algoritmo k-NN a problemas prácticos de clasificación y reconocer sus aplicaciones en escenarios del mundo real.

UTILIDAD
¿Por qué aprender sobre Aprendizaje Basado en Instancias?

Aplicaciones en el mundo real:

- Sistemas de recomendación (Amazon, Netflix, Spotify)
- Diagnóstico médico y análisis de historiales clínicos
- Reconocimiento de patrones en imágenes y texto
- Detección de fraudes en transacciones financieras
- Búsqueda y recuperación de información similar

¿Por qué aprender sobre Aprendizaje Basado en Instancias?

Ventajas profesionales:

- Habilidad demandada en roles de Ciencia de Datos y Machine Learning
- Fundamentos para entender sistemas de inteligencia artificial contemporáneos
- Base para técnicas más avanzadas como deep learning y sistemas de recomendación

En el contexto del curso:

- Conecta con los temas anteriores (árboles de decisión, reglas)
- Prepara para siguientes temas (clustering, optimización)
- Desarrolla habilidades prácticas en implementación de algoritmos ML

¿Por qué aprender sobre Aprendizaje Basado en Instancias?

¿Cómo se aplicará en el proyecto final?

- Posible uso para sistemas de recomendación básicos
- Clasificación de datos basada en similitud
- Solución de problemas de *pattern recognition*

Habilidad clave que desarrollarás:

"La capacidad de implementar y ajustar sistemas que aprenden directamente de los datos sin necesidad de construir modelos complejos predefinidos"

TRANSFORMACIÓN
Aprendizaje Basado en Instancias

Definición:
Los algoritmos de aprendizaje basado en instancias (o *instance-based learning*) aprenden directamente de las instancias/exemplos disponibles en los datos, sin construir un modelo general explícito.

Características clave:

- Almacenan los datos de entrenamiento en memoria.
- Las predicciones se hacen comparando nuevas instancias con las almacenadas.
- También se conocen como métodos **"perezosos"** (*lazy learning*), ya que el cálculo se realiza en tiempo de predicción.

Aspectos Prácticos del Aprendizaje Basado en Instancias Ventajas Y Desventajas.

Ventajas:

- Simple de implementar y entender.
- No requiere entrenamiento explícito (fase de "aprendizaje" rápida).
- Se adapta naturalmente a nuevos datos.

Desventajas:

- Costoso computacionalmente en predicción (alto consumo de memoria y CPU).
- Sensible a datos irrelevantes o ruidosos.
- Requiere preprocesamiento (normalización de datos).

Algoritmos representativos:

- k-Vecinos Más Cercanos (k-NN)
- Aprendizaje Basado en Casos (CBR)
- Máquinas de Vectores de Soporte (SVM) con kernels radiales

K-Vecinos Más Cercanos (K-NN)

El algoritmo K-NN es un método de aprendizaje supervisado basado en instancias que clasifica nuevos puntos según la mayoría de votos de sus k vecinos más cercanos.

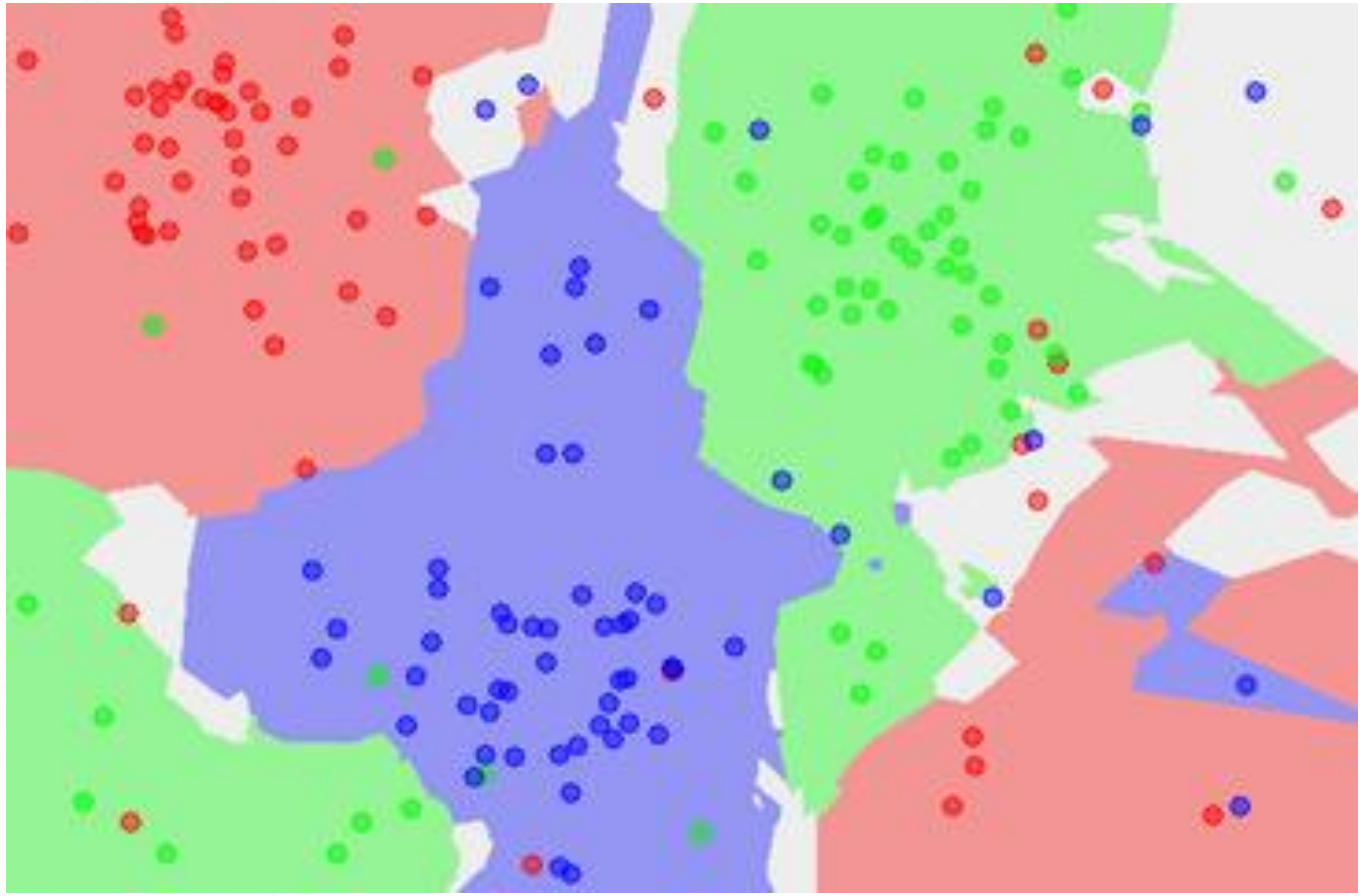
$$\text{Distancia Euclidiana: } d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

k-NN: Implementación

Cómo funciona k-NN:

- Almacena todas las instancias de entrenamiento.
- Para una nueva instancia, calcula la distancia a todas las instancias almacenadas.
- Selecciona las k instancias más cercanas.
- Asigna la clase mayoritaria (clasificación) o el promedio (regresión).

k-NN: Implementación



k-NN: Código de ejemplo

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

# Cargar datos
iris = load_iris()
X, y = iris.data, iris.target

# Dividir datos
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

k-NN: Código de ejemplo

```
# Entrenar modelo k-NN (k=3)
model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train, y_train)

# Predecir y evaluar
accuracy = model.score(X_test, y_test)
print(f"Precisión: {accuracy:.2f}")
```

Hyperparámetros clave:

- n_neighbors (k): Número de vecinos.
- metric: Distancia (euclidiana, manhattan, etc.).
- weights: Peso de los vecinos (uniforme o por distancia).

Ejercicio Práctico - k-NN con Scikit-Learn

Título: Laboratorio: Implementación de k-NN

Tareas:

- Cargar el dataset de cáncer de mama de Scikit-Learn.
- Normalizar los datos usando StandardScaler.
- Entrenar un modelo k-NN con k=5 y métrica euclidiana.
- Evaluar la precisión y matriz de confusión.
- Optimizar k usando validación cruzada.

Código de referencia:

```
from sklearn.datasets import load_breast_cancer
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV

# Cargar y normalizar datos
data = load_breast_cancer()
X, y = data.data, data.target
X_scaled = StandardScaler().fit_transform(X)

# Optimizar k
param_grid = {'n_neighbors': range(1, 15)}
knn = KNeighborsClassifier()
grid = GridSearchCV(knn, param_grid, cv=5)
grid.fit(X_scaled, y)

print("Mejor k:", grid.best_params_)
```

Aprendizaje Basado en Casos (CBR)
¿Qué es el CBR?

- Método de aprendizaje que resuelve nuevos problemas basándose en soluciones de casos pasados.
- Inspirado en cómo los humanos aprenden por experiencia.

Ejemplo práctico:

- Diagnóstico médico: Un médico recuerda un caso similar para diagnosticar una enfermedad.

Las 4 etapas del ciclo CBR:

- **Recuperar**: Buscar casos similares en la base de datos.
- **Reutilizar**: Adaptar la solución del caso similar al problema actual.
- **Revisar**: Evaluar si la solución propuesta funciona.
- **Retener**: Guardar el nuevo caso en la base de datos para futuro uso.

Aplicaciones:

- Asistentes virtuales de customer service.
- Sistemas de recomendación.
- Diagnóstico técnico o médico.

Implementación de CBR en Python

```
from sklearn.neighbors import NearestNeighbors
import numpy as np

# Base de datos de casos pasados (ejemplo: síntomas y diagnósticos)
casos = np.array([[1, 0, 1], [0, 1, 0], [1, 1, 1]])  # Síntomas
diagnosticos = ['Gripe', 'Alergia', 'COVID']  # Diagnósticos

# Nuevo caso a diagnosticar
nuevo_caso = np.array([[1, 0, 0]])

# Buscar caso más similar
modelo = NearestNeighbors(n_neighbors=1).fit(casos)
distancia, indice = modelo.kneighbors(nuevo_caso)

print(f"Diagnóstico sugerido: {diagnosticos[indice[0][0]]}")
```

SVM con Kernel Radial (RBF)
¿Qué es un Kernel Radial?

- Función que transforma datos no lineales a un espacio dimensional superior donde son linealmente separables.
- Fórmula:

$$K(x, y) = \exp\left(-\gamma \|x - y\|^2\right)$$

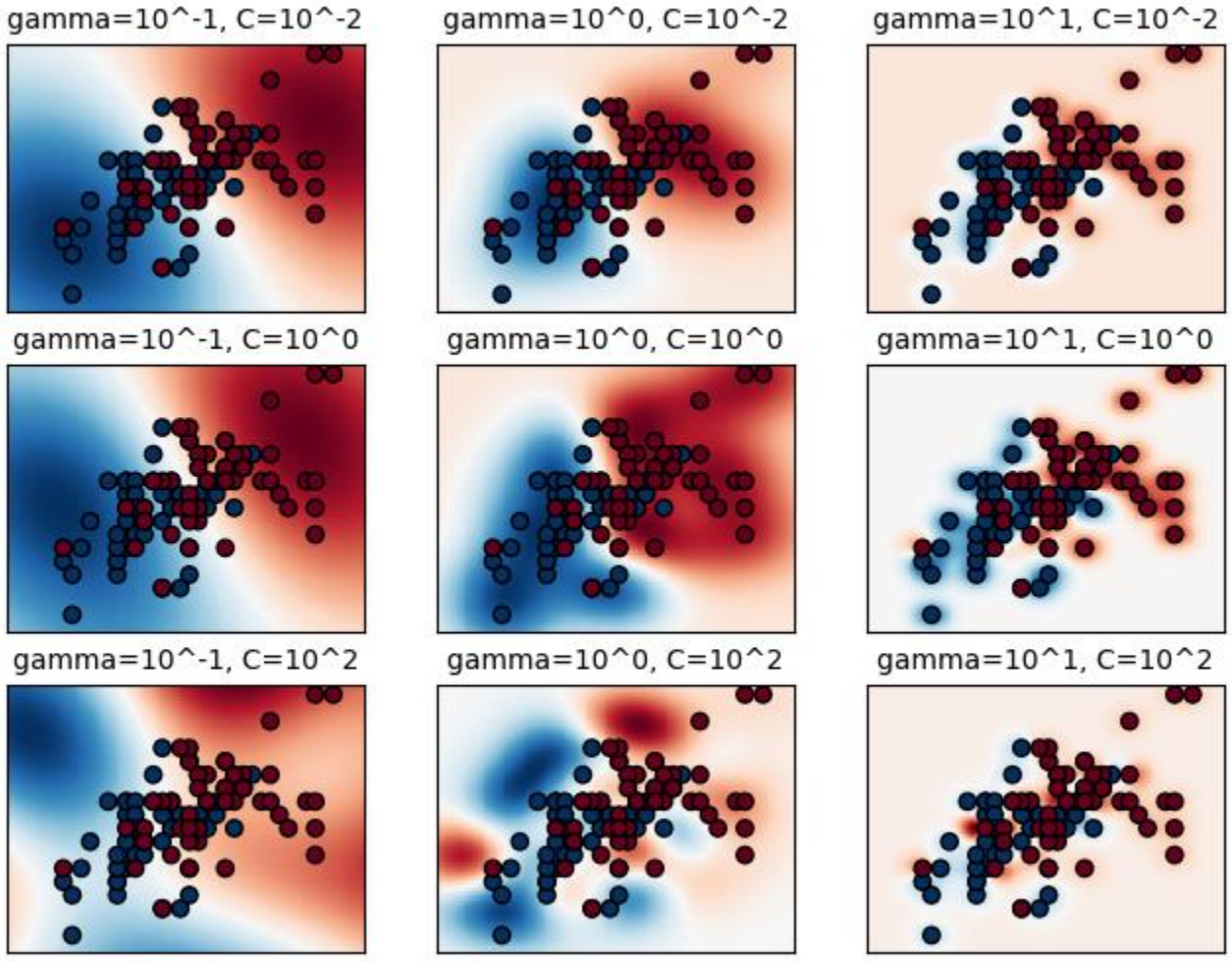
Ventajas:

- Maneja datos no lineales.
- Eficaz en problemas complejos (ej: reconocimiento de imágenes).

Hyperparámetros clave:

- C: Controla el trade-off entre sobreajuste y generalización.
- gamma: Define el alcance de la influencia de cada ejemplo.

SVM con Kernel Radial (RBF)



https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html

Implementación de SVM Radial en Python

```
from sklearn.svm import SVC
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

# Cargar datos
X, y = load_iris(return_X_y=True)

# Dividir datos
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

# Entrenar SVM con kernel radial
modelo = SVC(kernel='rbf', C=1.0, gamma='scale')
modelo.fit(X_train, y_train)

# Precisión
precision = modelo.score(X_test, y_test)
print(f"Precisión: {precision:.2f}")
```

Comparación de Métodos - KNN, CBR y SVM Radial

Característica	K-NN	CBR	SVM Radial
Tipo de aprendizaje	Basado en instancias (lazy)	Basado en casos (lazy)	Basado en modelos (eager)
Funcionamiento	Clasifica por votación de vecinos	Recupera y adapta casos similares	Transforma datos con kernel y busca margen máximo
Interpretabilidad	Media (depende de k)	Alta (explica por casos similares)	Baja (kernel como caja negra)
Coste computacional	Alto en predicción (almacena todos los datos)	Alto en recuperación (depende de la base de casos)	Alto en entrenamiento (optimización cuadrática)
Manejo de datos no lineales	Sí (con métricas de distancia)	Sí (con métricas de similitud)	Sí (especialidad del kernel radial)
Hyperparámetros clave	k, métrica de distancia	Métrica de similitud, número de casos	C, gamma
Aplicaciones típicas	Reconocimiento de patrones, recomendación	Diagnóstico, soporte técnico	Reconocimiento de imágenes, bioinformática
Ventajas principales	Simple de implementar	Explicable y basado en experiencia	Alta precisión en problemas complejos
Desventajas principales	Costoso con grandes volúmenes de datos	Requiere base de casos bien estructurada	Difícil interpretación y ajuste de parámetros

CIERRE
Conclusiones

Puntos Clave Consolidados:

- Los métodos basados en instancias (K-NN, CBR) aprenden directamente de los datos sin construir modelos explícitos
- Son ideales cuando la similitud entre casos es más importante que las reglas abstractas
- La elección de la métrica de distancia y el valor de k son críticos para el rendimiento de K-NN

Aportes Únicos:

- **Ventaja principal**: Adaptabilidad natural a nuevos datos y patrones
- **Fortaleza**: Explicabilidad basada en casos concretos y similares
- **Aplicación ideal**: Problemas donde la experiencia histórica es invaluable

Conclusiones

Limitaciones a Considerar:

- Alto costo computacional en predicción para grandes volúmenes de datos
- Sensibilidad a datos irrelevantes o ruidosos
- Requiere preprocesamiento cuidadoso (normalización, selección de características)

Próximo Paso Evolutivo:

- Del aprendizaje "perezoso" (instancias) al aprendizaje "activo" (modelos explícitos)
- Preparación para métodos más complejos como SVM y redes neuronales

Universidad
Tecnológica
del Perú