Clase 078 — SVM no lineal: kernel polinomial y RBF

Parte: 1 — Machine Learning Clásico · Fuente: Géron, cap. 5. Duración estimada: 70 min.

Clase 078 — SVM no lineal: kernel polinomial y RBF

> *Parte: 1 — Machine Learning Clásico · Fuente: Géron, cap. 5. Duración estimada: 70 min.*

Objetivo

Que el alumno entrene SVMs sobre datos no linealmente separables usando el kernel trick: en lugar de generar features polinómicas a mano (caro en memoria), SVC calcula el producto interno en el espacio expandido vía una función kernel. Foco en kernel polinomial y RBF (Gaussian), y cómo gamma y C controlan el bias-variance trade-off.

Resultados de aprendizaje

Al finalizar la clase, el alumno podrá:

1. Explicar el kernel trick: por qué K(x, x') evita materializar el feature map φ(x).
2. Entrenar SVC(kernel='poly') y SVC(kernel='rbf') en datasets no lineales (moons, circles).
3. Tunear gamma y C con GridSearchCV entendiendo el efecto en la frontera.
4. Elegir kernel según geometría del problema (polinomial vs RBF vs lineal).
5. Reconocer el límite computacional de SVC: complejidad entre O(n²) y O(n³), no escala a >100k filas.

Temas

#	Tema	Por qué importa
1	Datos no linealmente separables (moons, ci	Motivación: lineal no alcanza.
2	Polynomial features manuales vs kernel tri	Kernel evita explosión combinatoria.
3	Kernel polinomial: degree, coef0, gamma	Captura interacciones de orden d.
4	Kernel RBF (Gaussian): gamma como inverso	Default robusto en sklearn.
5	C (regularización) × gamma (forma)	Grid 2D clásico.
6	Complejidad O(n²)–O(n³) y LinearSVC / SGD	Cuándo NO usar SVC.

Definiciones y características

Kernel trick

: Truco matemático que reemplaza el producto interno φ(x)·φ(x') en el espacio expandido por una función K(x, x') calculable directamente en el espacio original. Permite trabajar en espacios de dimensión infinita (RBF) sin materializarlos. Requisito: K debe ser definida positiva (condición de Mercer).

Kernel RBF (Gaussian)

: K(x, x') = exp(-γ · ||x - x'||²). Mide similitud por distancia: 1 si x = x', → 0 cuando se alejan. Equivale a un feature map infinito-dimensional. Default razonable cuando no sabés qué kernel usar.

gamma (γ)

: Inverso del ancho del kernel RBF. γ alto → cada punto influye solo en su vecindad inmediata → frontera muy ondulada → overfitting. γ bajo → influencia amplia → frontera suave → underfitting. En sklearn:

gamma='scale' (default) = 1 / (n_features · X.var()).

Kernel polinomial

: K(x, x') = (γ · x·x' + coef0)^degree. Captura interacciones hasta orden degree. coef0 controla el peso de los términos de orden alto vs bajo. Típicamente degree=2 o 3; más alto suele overfittear.

Kernel sigmoid

: K(x, x') = tanh(γ · x·x' + coef0). Rara vez supera a RBF en la práctica; se mantiene por razones históricas (similitud con redes neuronales).

Complejidad O(n²)–O(n³)

: SVC resuelve un QP cuadrático sobre los n ejemplos. En la práctica, libsvm escala entre O(n²) y O(n³) según la fracción de support vectors. Para >50k–100k filas: usar LinearSVC (si es lineal) o SGDClassifier(loss='hinge').

Condición de Mercer

: Para que una función K sea un kernel válido, su matriz de Gram $[K(x_i, x_j)]$ debe ser semidefinida positiva para cualquier conjunto finito de puntos. RBF, polinomial y lineal la cumplen; sigmoid solo bajo ciertos parámetros.

C (regularización)

: Inverso del parámetro de regularización L2. C alto → poca tolerancia a violaciones del margen → margen estrecho, posible overfit. C bajo → margen amplio, más violaciones permitidas, modelo más simple.

Dataset / recursos

- sklearn.datasets.make_moons(n_samples=500, noise=0.15) — clásico no lineal.
- sklearn.datasets.make_circles(n_samples=500, noise=0.1, factor=0.4) — radial puro, ideal para mostrar RBF.

Ejercicios

1. Lineal falla. Entrená SVC(kernel='linear') sobre make_moons. Reportá accuracy y graficá la frontera. Mostrá visualmente que es inadecuada.

2. Polynomial kernel. SVC(kernel='poly', degree=3, coef0=1, C=5) sobre moons. Comparar accuracy y forma de la frontera vs lineal.

3. RBF kernel. SVC(kernel='rbf', gamma=5, C=1) sobre circles. Variar gamma ∈ {0.1, 1, 10, 100} con el mismo C y graficar las 4 fronteras lado a lado.

4. Grid search 2D. GridSearchCV sobre gamma ∈ {0.01, 0.1, 1, 10} × C ∈ {0.1, 1, 10, 100} con cv=5 sobre moons. Reportar el mejor par y matriz de scores como heatmap.

5. Pipeline con StandardScaler. SVMs son sensibles a escala. Armá Pipeline([('sc', StandardScaler()), ('svc', SVC(kernel='rbf'))]) y comparar accuracy con y sin scaler en un dataset con features de magnitudes muy distintas.

Homework verificable

Notebook con make_moons(n_samples=1000, noise=0.2). Train/test split 80/20. Entrenar tres modelos: (a)

SVC(kernel='linear'), (b) SVC(kernel='poly', degree=3), (c) SVC(kernel='rbf') con gamma y C tuneados vía GridSearchCV(cv=5). Reportar accuracy en test de los tres y graficar las tres fronteras de decisión en una grilla 1×3.

Criterio de aceptación: El modelo RBF tuneado supera al lineal en al menos +15 puntos de accuracy. El notebook incluye el heatmap de GridSearchCV y las tres fronteras visualizadas.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
Usar SVC en datasets >100k filas y que el	Complejidad O(n²)–O(n³). Fix: si el proble
Accuracy pésima sin escalar features	RBF y polinomial dependen de distancias. U
gamma alto da 100% en train y 60% en test	Overfitting clásico: γ alto = frontera ond
SVC(kernel='poly', degree=10) tarda eterni	Polinomios de alto grado explotan numérica
No setear random_state y resultados irrepr	SVC con probability=True o algunos solvers

Preguntas frecuentes

¿RBF o polynomial?

Si no sabés nada del problema: RBF (default robusto, un solo hiperparámetro relevante gamma). Polynomial conviene si tenés razones para creer que las interacciones son de orden bajo y fijo (ej. features físicas que se multiplican). RBF tiende a ganar en benchmarks tabulares.

¿Qué pasa si γ → 0 en RBF?

El kernel tiende a constante → todos los puntos parecen iguales → modelo equivale a clasificar por mayoría. Frontera trivial (clase mayoritaria en todo el espacio).

¿Cuántos support vectors es "muchos"?

Si len(svm.support_) > 0.5 · n_train, el modelo está overfitteando o C es muy alto. SVMs eficientes tienen pocos SVs (los del borde).

¿Por qué gamma='scale' y no 'auto'?

'scale' = 1 / (n_features · X.var()) — tiene en cuenta la varianza de los datos, robusto a escala. 'auto' = 1 / n_features (default viejo, deprecado conceptualmente). Usá 'scale' o un valor explícito tuneado.

¿Puedo usar SVM kernelizada con 1M de filas?

No con SVC. Alternativas: (a) LinearSVC si el problema es separable linealmente tras feature engineering; (b) Nystroem + LinearSVC para aproximar RBF en O(n); (c) SGDClassifier(loss='hinge'); (d) cambiar de modelo (Gradient Boosting suele ganar en tabular grande).

Referencias

- Géron, cap. 5 — Nonlinear SVM Classification y The Kernel Trick.
- sklearn — SVC
- sklearn — RBF kernel intuition
- sklearn — Kernel approximation (Nystroem)

Siguiente clase

Clase 079 — SVM para regresión (SVR)

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb