Clase 092 — PCA: proyección, varianza explicada, incremental, randomized, kernel

Parte: 1 — Machine Learning Clásico · Fuente: Géron, cap. 8. Duración estimada: 80 min.

Clase 092 — PCA: proyección, varianza explicada, incremental, randomized, kernel

Parte: 1 — Machine Learning Clásico · Fuente: Géron, cap. 8. Duración estimada: 80 min.

Objetivo

Dominar PCA como técnica de reducción de dimensionalidad lineal: entender la proyección al subespacio de máxima varianza vía SVD, elegir el número de componentes con explained_variance_ratio_, y aplicar las variantes (Incremental, Randomized, Kernel) según el tamaño y la geometría del dataset.

Resultados de aprendizaje

Al finalizar la clase, el estudiante podrá:

- Explicar geométricamente qué hace PCA (proyección al hiperplano que maximiza la varianza).
- Ajustar PCA de scikit-learn y leer components_, explained_variance_ratio_ y singular_values_.
- Elegir el número de componentes vía umbral de varianza acumulada (ej. 95%) o codo del scree plot.
- Decidir entre PCA, IncrementalPCA y PCA(svd_solver="randomized") según memoria y velocidad.
- Aplicar KernelPCA (RBF/poly) para datasets con estructura no-lineal y tunear gamma con GridSearchCV.

Temas

1. Maldición de la dimensionalidad y motivación de PCA.
2. Proyección al subespacio de máxima varianza: intuición geométrica.
3. SVD como motor algebraico de PCA.
4. Varianza explicada y elección de n_components (entero, ratio, "mle").
5. IncrementalPCA para datasets que no entran en RAM (partial_fit).
6. Randomized PCA (svd_solver="randomized") para acelerar en alta dimensión.
7. KernelPCA con kernels RBF, polinómico y sigmoide; tuning de gamma.
8. Pipeline completo: StandardScaler → PCA → modelo.

Definiciones y características

- PCA (Principal Component Analysis): técnica lineal que proyecta los datos sobre los ejes ortogonales que capturan máxima varianza.
- Componentes principales: vectores ortonormales (filas de components_) que definen el nuevo sistema de coordenadas; el primero apunta a la dirección de máxima varianza, el segundo a la siguiente ortogonal, etc.
- SVD (Singular Value Decomposition): factorización $X = U \Sigma V^T$; las columnas de V son los componentes principales y $\Sigma^2/(n-1)$ da la varianza explicada.
- explained_variance_ratio_: array con la fracción de varianza total capturada por cada componente; su suma acumulada guía la elección de n_components.
- IncrementalPCA: versión que procesa mini-batches con partial_fit; útil cuando X no entra en memoria. Costo: ligeramente más lenta y aproximada.

- Randomized PCA (svd_solver="randomized"): algoritmo estocástico que encuentra los primeros d componentes en O(m·d²) + O(d³) en vez de O(m·n²) + O(n³). Default cuando n_components << min(m, n).
- KernelPCA: aplica PCA en el espacio de features inducido por un kernel (RBF, poly, sigmoid); permite capturar estructura no-lineal (ej. swiss roll).
- gamma: hiperparámetro del kernel RBF que controla el ancho de la gaussiana; valor chico = kernel suave, valor alto = kernel angosto. Se tunea con GridSearchCV.
- Scaling previo: PCA es sensible a la escala — siempre estandarizar (StandardScaler) antes, salvo que todas las variables ya estén en la misma unidad.

Dataset / recursos

- MNIST (fetch_openml("mnist_784")) — 70.000 × 784, ideal para mostrar reducción a ~150 componentes preservando 95% de varianza.
- Swiss roll (sklearn.datasets.make_swiss_roll) — para contrastar PCA lineal vs KernelPCA RBF.
- Géron, Hands-On ML, cap. 8 — Dimensionality Reduction, sección "PCA".

Ejercicios

1. Cargá MNIST, aplicá StandardScaler + PCA(n_components=0.95) y reportá cuántos componentes quedaron.
2. Graficá la curva de varianza acumulada (cumsum(explained_variance_ratio_)) y marcá el codo.
3. Compará tiempos de PCA(svd_solver="full") vs "randomized" sobre MNIST con %timeit.
4. Usá IncrementalPCA con n_batches=100 y verificá que el resultado se aproxima al PCA full (cosine similarity entre componentes > 0.99).
5. Sobre make_swiss_roll(n_samples=1000), aplicá KernelPCA(kernel="rbf", gamma=0.04, n_components=2) y comparalo con PCA lineal en un scatter 2D.

Homework verificable

Construí un pipeline StandardScaler → PCA → LogisticRegression sobre MNIST (subset 10k) y:

1. Reportá accuracy con n_components=50, 100, 154 (154 ≈ 95% varianza).
2. Mostrá la matriz pipe.named_steps["pca"].components_.shape y el explained_variance_ratio_.sum().
3. Entregá notebook + un párrafo explicando el trade-off accuracy vs nº de componentes.

Criterio de aceptación: accuracy ≥ 0.90 con n_components=100 y curva de varianza acumulada graficada.

Errores comunes

1. No escalar antes de PCA — variables con varianza grande (por unidad, no por información) dominan los componentes. Siempre StandardScaler primero.
2. Usar PCA para reducir dimensionalidad antes de un modelo basado en árboles (RF, XGBoost) — no aporta, los árboles son invariantes a rotaciones de features.
3. Interpretar los componentes como "features originales seleccionadas" — son combinaciones lineales, no subconjuntos.
4. Aplicar PCA().fit(X_train_and_test) — fuga de datos. Hacer fit solo sobre train y transform sobre test.
5. Olvidar que KernelPCA no tiene inverse_transform por default — hay que pasar fit_inverse_transform=True.

Preguntas frecuentes

1. ¿Cuántos componentes elegir? Regla práctica: el menor d tal que cumsum(explained_variance_ratio_) ≥ 0.95. También se puede usar n_components="mle" (Minka) o ver el codo del scree plot.
2. ¿PCA mejora siempre la accuracy? No. Reduce ruido y acelera el entrenamiento, pero puede perder información discriminativa. Validá con CV.
3. ¿Cuándo usar Incremental vs Randomized? Incremental si X no entra en RAM; Randomized si entra pero querés velocidad y n_components es chico vs dimensión original.
4. ¿KernelPCA reemplaza a t-SNE/UMAP? No. KernelPCA preserva varianza global en el espacio kernel; t-SNE/UMAP preservan estructura local. Son herramientas distintas (ver clases 083-084).
5. ¿Por qué PCA es sensible a outliers? Porque maximiza varianza y los outliers tienen varianza desproporcionada. Considerá RobustScaler o TruncatedSVD si hay sparse + outliers.

Referencias

- Géron, A. Hands-On Machine Learning, 3ª ed., cap. 8 — Dimensionality Reduction, secciones "PCA", "Randomized PCA", "Incremental PCA", "Kernel PCA".
- scikit-learn user guide: <https://scikit-learn.org/stable/modules/decomposition.html#pca>
- Halko, Martinsson & Tropp (2011), Finding structure with randomness — paper original de Randomized SVD.
- Schölkopf, Smola & Müller (1998), Nonlinear component analysis as a kernel eigenvalue problem.

Siguiente clase

Clase 093 — LLE (Locally Linear Embedding)

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb