Introduction to Machine Learning - Challenge 2

Autore: Bredariol Francesco

Data di consegna: April 9, 2025

1 Introduzione

In questa challenge si testeranno come alcuni metodi di regressione e di clustering classici migliorino una volta kernelizzati. I metodi presi in considerazione saranno Ridge Regression e Pca. I dataset sono generati sinteticamente tramite funzioni ad hoc.

2 Kernel Ridge Regression

Questa parte è dedicata alla Ridge Regression ed alla sua controparte kernelizzata.

2.1 Dataset

Il dataset è generato dalla seguente funzione:

$$\epsilon \sim N(0, 1) \tag{1}$$

$$y = (x + 4)(x + 1)(\cos(x) - 1)(x - e) + \epsilon \tag{2}$$

Che visualmente risulta essere:

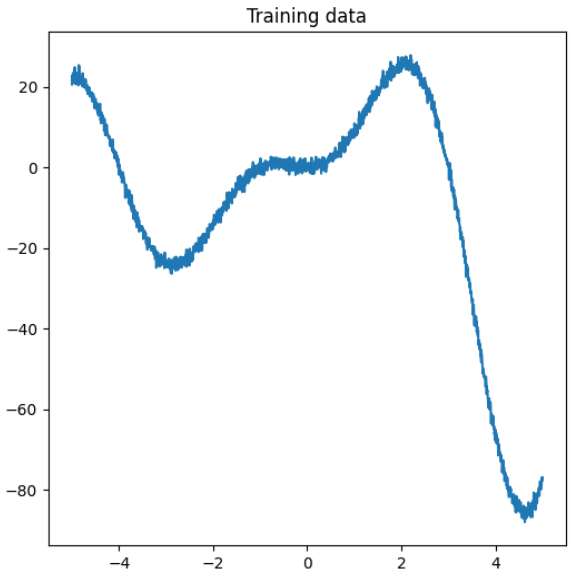

Figure 1: Visualizzazione di 2 nell'intervallo [-5, 5].

2.2 Ridge vs Kernel Ridge

I risultati della semplice Ridge regression lasciano a desiderare, come d'altronde ci si poteva aspettare dal momento in cui il dataset è fortemente non lineare. La Kernel Ridge regression funziona molto bene usando il kernel gaussiano, mentre con il kernel polinomiale non molto.

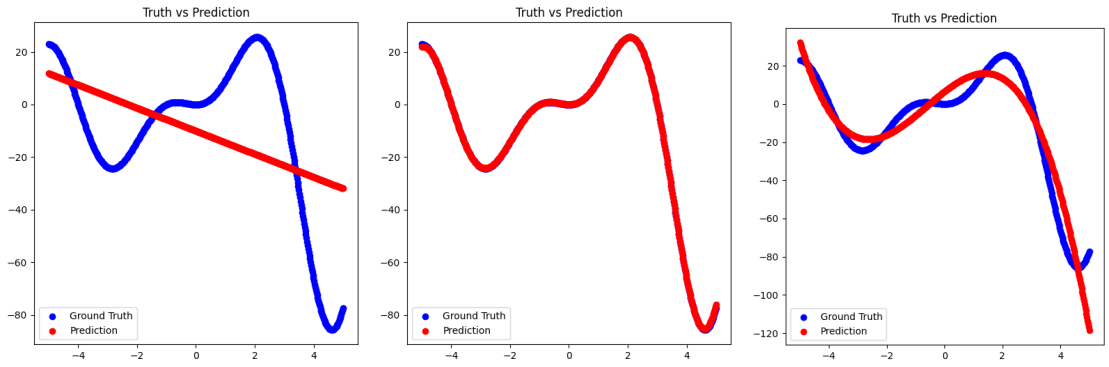

Figure 2: Migliori risultati ottenuti (in termini di MSE) da vari metodi di Ridge Regression. Panel SX : Ridge Regression; Panel Centrale : Rbf Kernel Ridge Regression; Panel DX : Polynomial Kernel Ridge Regression. Nei vari plot in blu è rappresentata la ground truth mentre in rosso la previsione ottenuta usando il relativo metodo di regressione. Si può decisamente notare come il Kernel Gaussiano abbia praticamente approssimato perfettamente la funzione target.

Altre esplorazioni in termini di parametri sono state eseguite tramite Grid Search, riportiamo i risultati più rilevanti ottenuti.

Kernel	Degree	Gamma	Alpha	MSE	R^2
None	1	-	1	716.167	0.18
Rbf	-	1	1	0.055	0.99
Poly	3	-	1	84.555	0.90

3 Kernel PCA

3.1 Dataset

Due diversi tipi di dataset sono stati testati. Il primo è generato tramite la funzione "make_circles" di sklearn, il secondo tramite la funzione "make_classification" di sklearn. Queste funzioni permettono di scegliere a priori la dimensionalità dei dati. Per una intuizione geometrica mostreremo dei dati in dimensione 2. Tuttavia nell'esercitazione PCA e Kernel PCA sono state utilizzate su dataset ad alta dimensionalità (d = 10+).

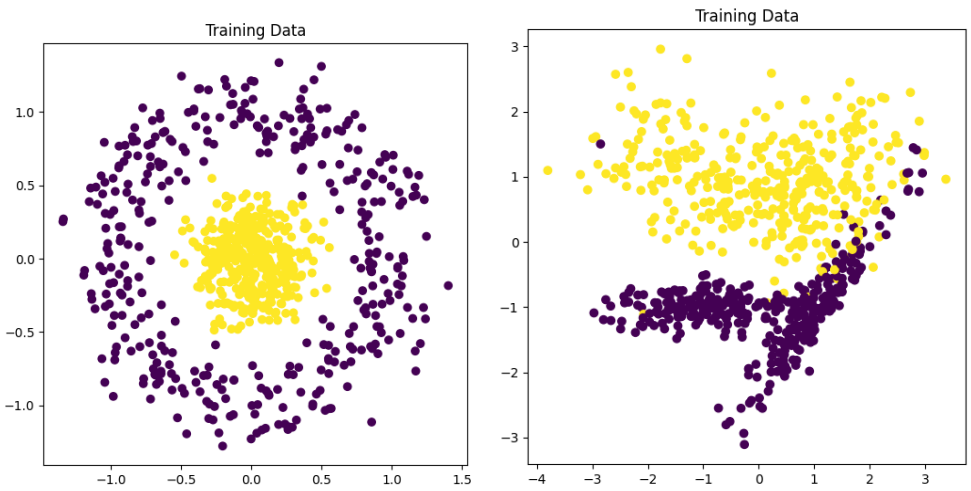

Figure 3: Visualizzazione 2 dimensionale dei dati di training. Panel SX : dati ottenuti dalla funzione make_circles; Panel DX : dati ottenuti dalla funzione make_classification. Si noti bene che questo è solo un esempio utile a comprendere intuitivamente la natura dei dati in alta dimensione su cui sono state successivamente applicate PCA e Kernel PCA.

3.2 PCA vs Kernel PCA

Per valutare l'efficacia di PCA e di Kernel PCA è stata valutata l'accuracy di una SVM lineare (medesimi parametri per entrambe le valutazioni) su i dati trasformati dai metodi di riduzione di dimensionalità.

3.2.1 Make_circles

Questa sottosezione è relativa ai dati generati da make_circles. I parametri usati per make_circles sono i seguenti :
n_samples=1000, noise=0.15, factor=0.2, random_state = 0.
Il kernel gaussiano ha portato ai migliori risultati.

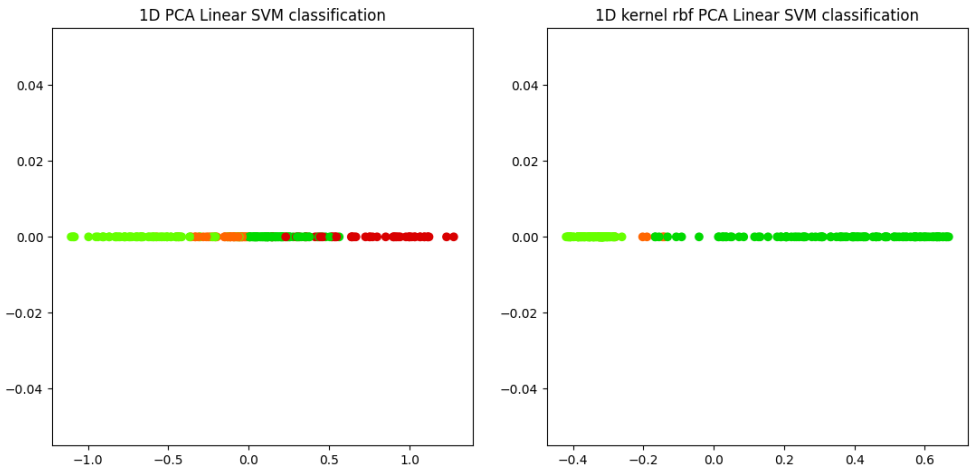

Figure 4: Risultati ottenuti dalla classificazione (SVM lineare) effettuata sulle proiezioni 1-dimensionali prodotte da PCA e Kernel PCA. Panel SX : risultati su PCA; Panel DX : risultati su Rbf Kernel PCA. I punti verdi e rossi indicano rispettivamente i punti classificati correttamente ed erroneamente dalla SVM. In base alla luminosità (verde chiaro/rosso chiaro, verde scuro/rosso scuro) si può individuare il tipo di errore effettuato.

Riportiamo in tabella i risultati ottenuti dai metodi PCA / Kernel PCA sotto forma di metriche relative alla SVM lineare che classifica i dati sulle proiezioni.

Dimensione	Kernel	Gamma	Accuracy
1	-	-	0.62
1	Rbf	10	0.99

3.2.2 Make_classification

Questa sottosezione è relativa ai dati generati da make_classification. I parametri usati per make_classification sono i seguenti :
n_samples = 1000, n_classes=2, n_features=10, n_informative=6, n_redundant=3, n_repeated=1, random_state = 0.
L'assenza di kernel ha portato ai migliori risultati.
Sono state effettuate ricerche in grid search per ottimizzare i risultati e ciò che ne è uscito è che i kernel (sia polinomiali che gaussiani) in questo particolare dataset performano male. Leggendo la documentazione del metodo make_classification si comprende che dovrebbe esserci un'innata linearità nei dati e dunque la semplice PCA per costruzione può catturare meglio questa struttura e ottenere buoni risultati. Riportiamo in tabella il risultato della semplice PCA contro il miglior risultato tra tutti i kernel testati.

Dimensione	Kernel	Degree	Accuracy
3	-	-	0.84
3	Poly	1	0.74