Machine Learning Challenge 2 Report

Dino Meng [SM3201466]

Introduzione

Keywords: Kernel methods, Kernel PCA, Kernel Ridge Regression, Grid Search

L'obbiettivo di questo report è quello di esplorare le potenzialità dei *kernel methods*, applicandoli a problemi di natura diversa.

Useremo dei dataset generati artificialmente che vanno a rappresentare problemi di tipologie diverse, tra cui la regressione, riduzione della dimensionalità e la classificazione.

??? #TODO

Metodologia

In questa sezione si descrivono i passaggi svolti per questo progetto.

Dataset 1: Regressione

Nel primo problema si genera un dataset a due variabili, di cui una è esplanatoria (i.e. indipendente) e l'altra è la variabile target. Denoteremo queste variabili rispettivamente con x, y.

In particolare, definiremo $y = f(x)$ sull'intervallo $x \in [-5, 5]$. La funzione da imparare generata è la seguente funzione non-lineare:

$$f(x) = (x+4)(x+1)(\cos x - 1)(x-3) + \varepsilon_x \tag{1}$$

Dove ε_x è il *noise*, generato casualmente seguendo la distribuzione normale $\varepsilon_x \sim \mathcal{N}(0, 1)$. Un *plot* rappresentativo di questa funzione è fornita in figura 1.

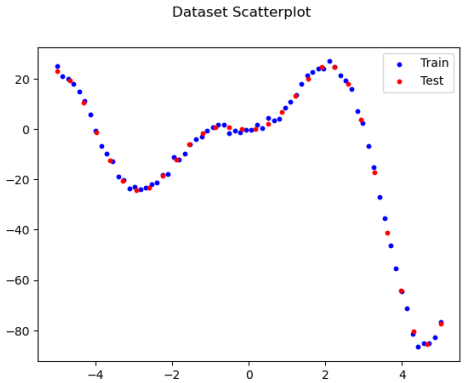

Figure 1: Non-linear Dataset

In totale abbiamo generato 100 punti per il dataset. Abbiamo diviso il dataset in due parti, quello di *training* e quello di *testing*; per la divisione dei dati abbiamo seguito la proporzione 70-30.

Per valutare i modelli, useremo la metrica dell'errore medio quadratico (*MSE*) e il punteggio *R2* sul dataset del *testing*.

Come primo approccio abbiamo addestrato e valutato una *ridge regression* non kernelizzata, fornendoci una specie di *baseline* per i modelli successivi.

Dopodiché abbiamo iniziato a sperimentare con varie *ridge regression* kernelizzate, facendo variare i suoi iperparametri: in particolare abbiamo usato il kernel gaussiano (*RBF*) e polinomiale.

Per trovare la migliore combinazione dei iperparametri, abbiamo effettuato due *Grid Search* sul dataset di allenamento. Per i modelli kernel abbiamo definito lo seguente spazio degli iperparametri

Table 1: Hyperparameters for GridSearchCV

Kernel	Parametro	Spazio
Gaussiano	`gamma`	$10^{-5}, \ldots, 10^{5}$
Polinomiale	`degree`	$2, 3, \ldots, 7, 8$
	`alpha`	$10^{0}, \ldots, 10^{4}$

Infine, per scegliere il miglior modello, abbiamo valutato il migliore modello gaussiano e polinomiale sul dataset di testing.

Dataset 2: Riduzione di Dimensionalità e Classificazione

Nel problema successivo si affronta uno dei problemi di classificazioni più noti - e quasi tipica per i metodi kernel: il dataset a due dimensioni e a due classi, disposte su due cerchi concentrici (fig. 2).

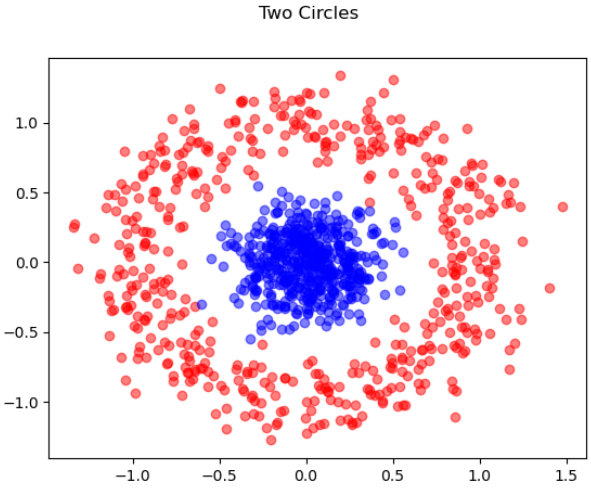

Figure 2: Two Circles

Prima di tutto, abbiamo applicato l'analisi delle componenti principali (*PCA*) sul dataset e proiettando la trasformazione sia in due che una dimensione. Dopodiché, abbiamo *fittato* una macchina a supporto vettoriale soft-margin (*SVM*) non kernelizzato sul dataset del training, e l'abbiamo valutato per fornire una *baseline* per i modelli successivi.

Dopodiché abbiamo applicato la *PCA* kernelizzata usando il kernel gaussiano col parametro $\gamma = 3$, e per verificare il ben-funzionamento del kernel scelto abbiamo addestrato e valutato una *SVM* kernelizzata col kernel selezionato.

Dataset 3: Riduzione di Dimensionalità e Classificazione (2)

Per l'ultimo problema abbiamo generato un dataset con la funzione `make_classification()` di Scikit-Learn (modulo `datasets`), che va a generare un problema di classificazione a due classi con 20 variabili.

Abbiamo dunque trasformato il dataset con la *PCA* e abbiamo preso la sua proiezione in due e tre dimensioni per dare una semplice visualizzazione dei dati. Dopodiché, abbiamo allenato e valutato una *SVM* sul dataset, per ottenere delle *performance baseline*.

Dopodichè abbiamo allenato e valutato più *SVM kernel* sul dataset, fornendo come parametri quelli di default su Scikit-Learn.

Per determinare quale fosse la miglior scelta del kernel con la migliore combinazione di iperparametri, abbiamo deciso di effettuare una ricerca casuale (*randomized search*) valutata mediante la convalida incrociata. La scelta di una ricerca casuale rispetto ad una *Grid Search* è motivata dal fatto che stiamo effettuando una ricerca su un dominio di iperparametri piuttosto ampio, infatti tre parametri ricevono numeri continui.

In particolare, abbiamo definito il dominio degli iperparametri della ricerca casuale con le seguenti variabili aleatorie (o liste, nel caso discreto):

Table 2: Hyperparameters for RandomSearchCV

Iperparametro	Tipo	Dominio*
`C`	continuo	$\mathcal{U}(-5\log(10), 5\log(10))$
`kernel`	categorico	poly, rbf, linear, sigmoid
`gamma`	continuo	$\mathcal{U}(-5\log(10), 5\log(10))$
`degree`	intero	2,3,4,5,6,7,8
`coef0`	continuo	$\mathcal{U}(-10, 10)$

* $\mathcal{U}(a, b)$ denota la distribuzione uniforme nell'intervallo $[a, b]$.

Risultati

Dataset 1

Riportiamo le prestazioni di tutti i modelli addestrati e valutati (sul test dataset), informato tabulare:

Table 3: Models Performances on Dataset 1

Kernel	**Valutazione**	**Punteggio**
Gaussiano (RBF)	R2	0.9810
	MSE	17.7460
Polinomiale	R2	0.8771
	MSE	114.7043
Lineare (Ridge Regression)	R2	0.2263
	MSE	772.0261

Plottiamo inoltre le predizioni del miglior modello, fornendoci un'idea grafica (fig. 3)

Infine, riportiamo i grafici delle predizioni dei modelli kernel con la variazione dei loro iperparametri (fig. 4, 5). Precisiamo che per la variazione del kernel polinomiale (fig. 5), la variazione dell'iperparametro `alpha` viene fatta variare col grado del polinomio: la variazione di `alpha` viene rappresentato da sfumature diverse del rosso.

Dataset 2

Riportiamo innanzitutto le rappresentazioni grafiche delle trasformazioni fornite dalla PCA, una di cui non-kernel e l'altra kernelizzata (fig. 6, 7).

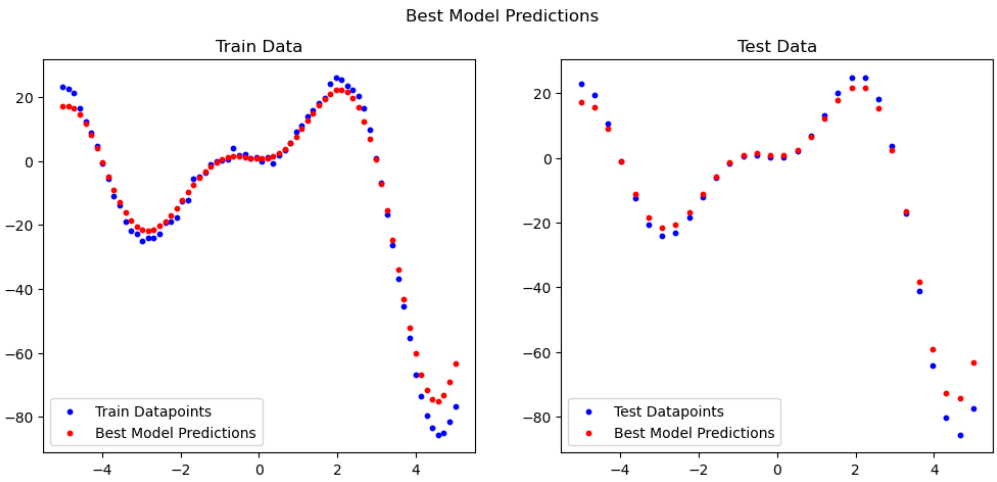

Figure 3: Best Model Predictions

Infine, i punteggi F1 (macro media) dei modelli SVM sono le seguenti:

Table 4: Models Performances on Dataset 2

Modello	**Train**	**Test**
SVM non kernel	0.65	0.59
SVM kernel	1.00	0.99

Dataset 3

Si plotta innanzitutto le proiezioni in due e tre dimensioni della PCA sul dataset (fig. 8)

Dopodiché riportiamo dalle varie trasformate fatte dalla kernel PCA. Le proiezioni in due e tre dimensioni sono riportate in figure separate (fig. 9, 10)

Adesso riportiamo le performance dei modelli addestrati sul dataset, usando la F1 score (macro media).

Table 5: Models Performances on Dataset 3

Modello	**Train**	**Test**
SVM non kernel	1.00	0.96
SVM kernel gaussiano	1.00	0.92
SVM kernel polinomiale	1.00	0.96
SVM kernel lineare	1.00	0.96
SVM kernel sigmoide	0.97	0.92
SVM Random Search	1.00	0.96

Infine, enunciamo che il miglior modello trovato dalla ricerca casuale è il **kernel PCA col kernel lineare**. Imperocchè il kernel è lineare, gli altri iperparametri sono irrilevanti e dunque omessi.

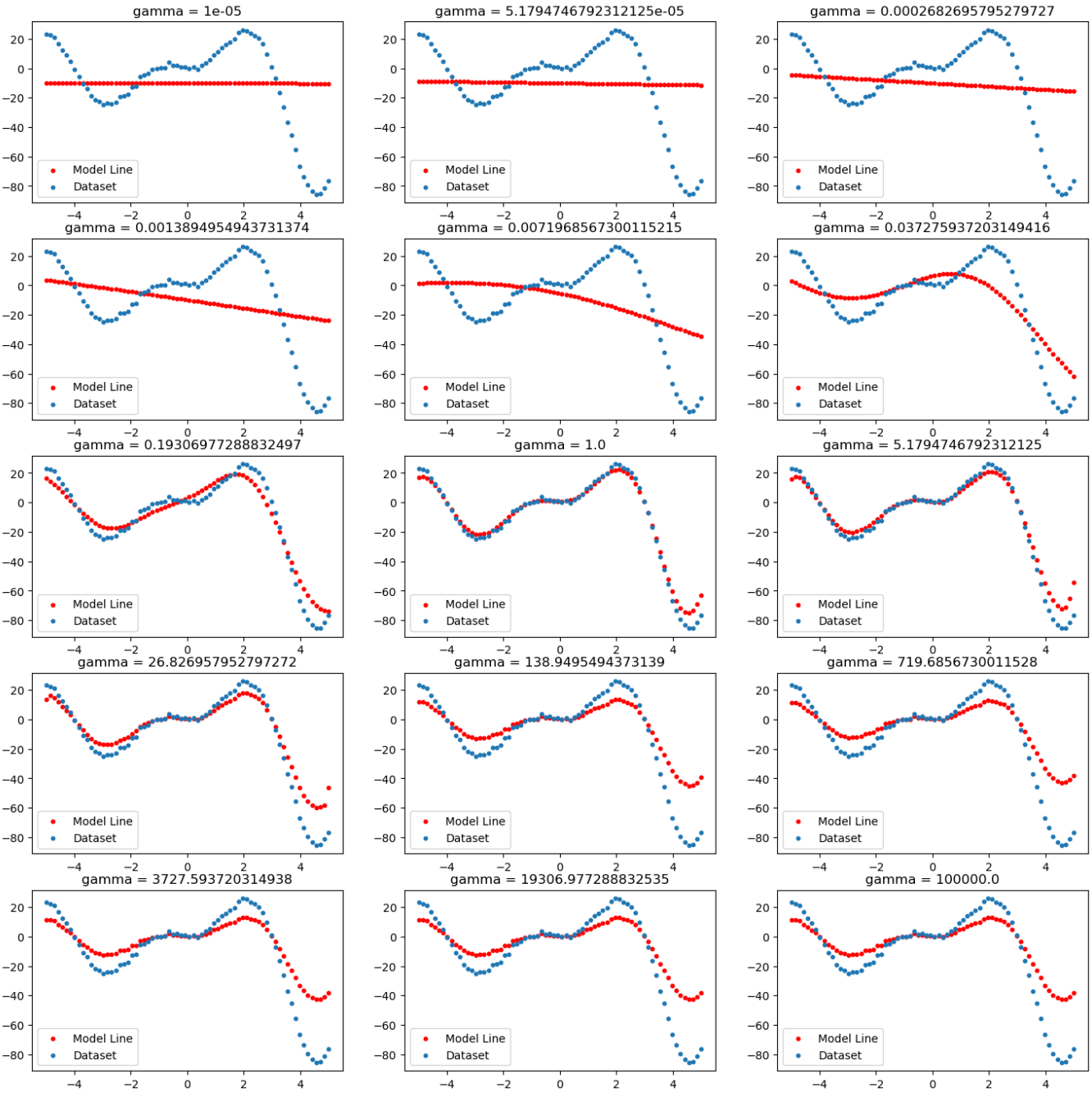

Figure 4: RBF Kernel Variations

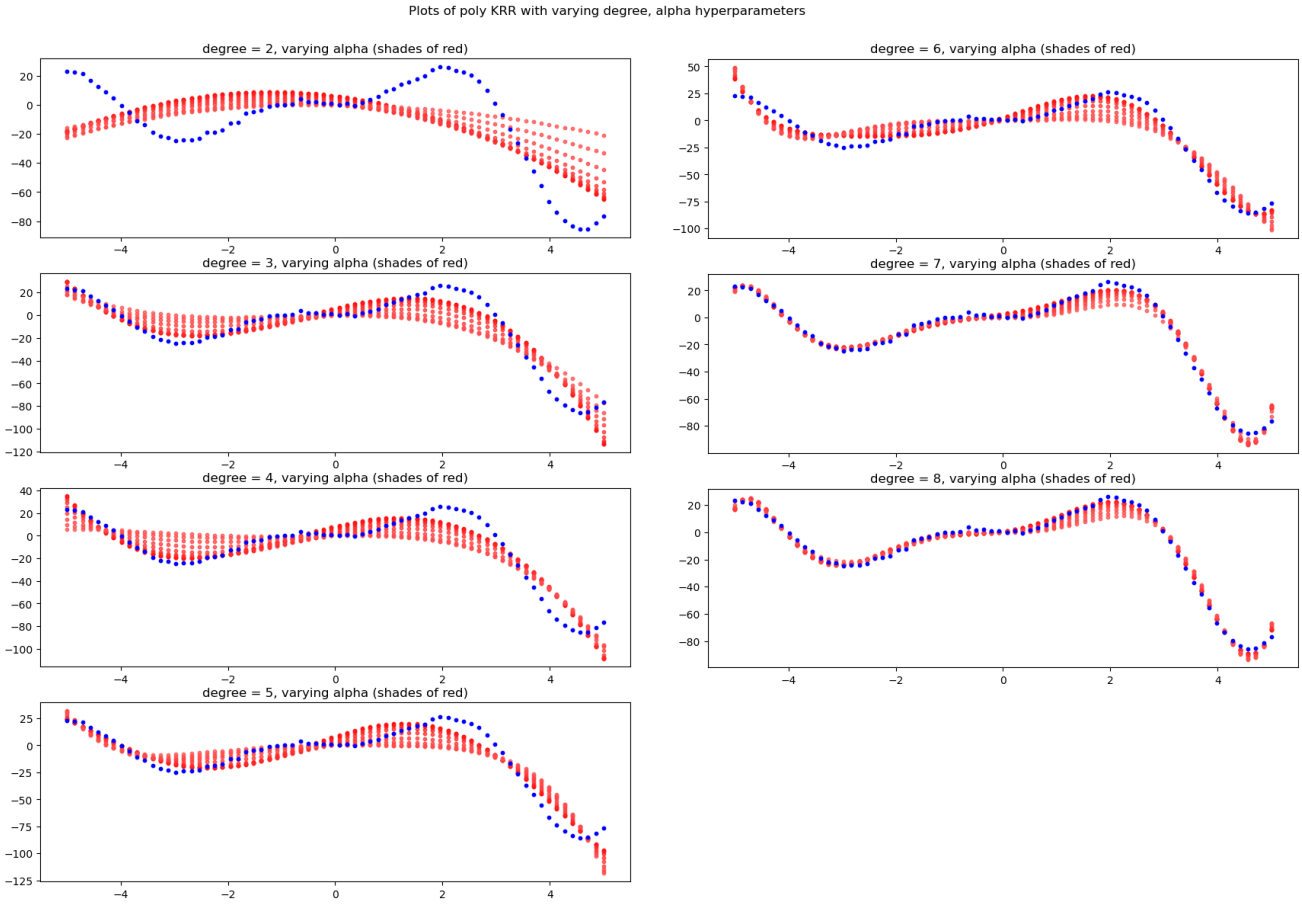

Figure 5: Polynomial Kernel Variations

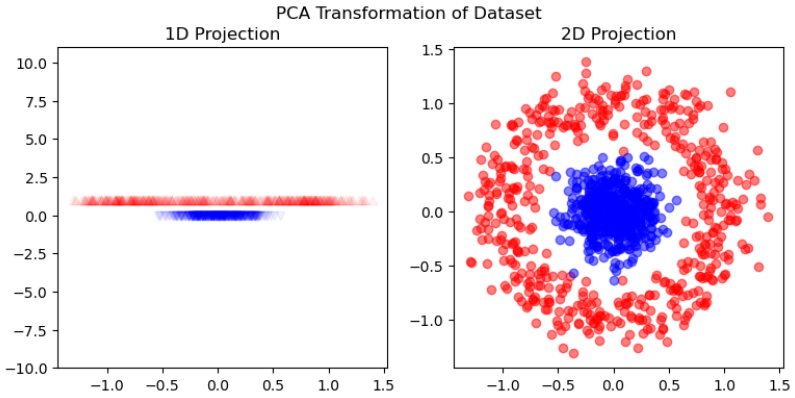

Figure 6: PCA Projections

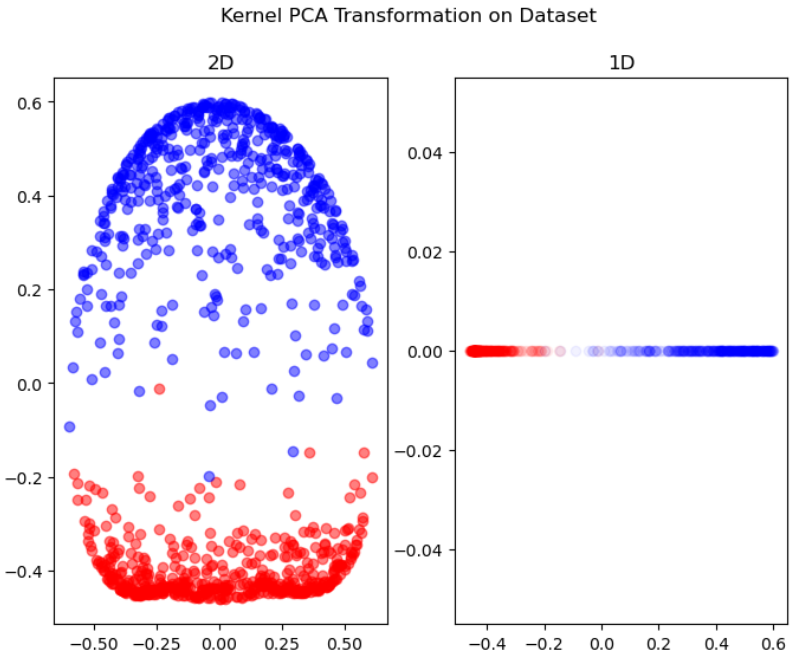

Figure 7: KPCA Projections

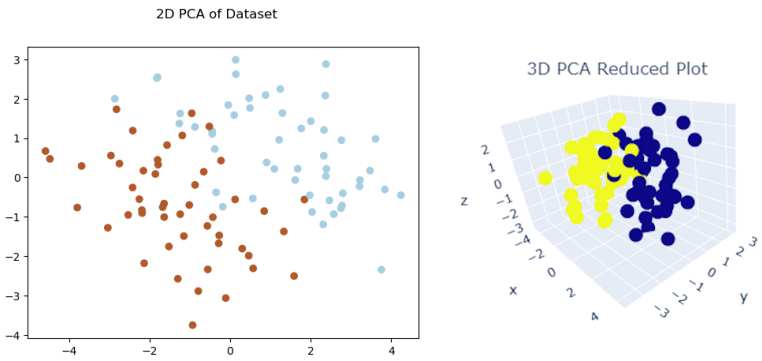

Figure 8: PCA Transformation in 2D, 3D

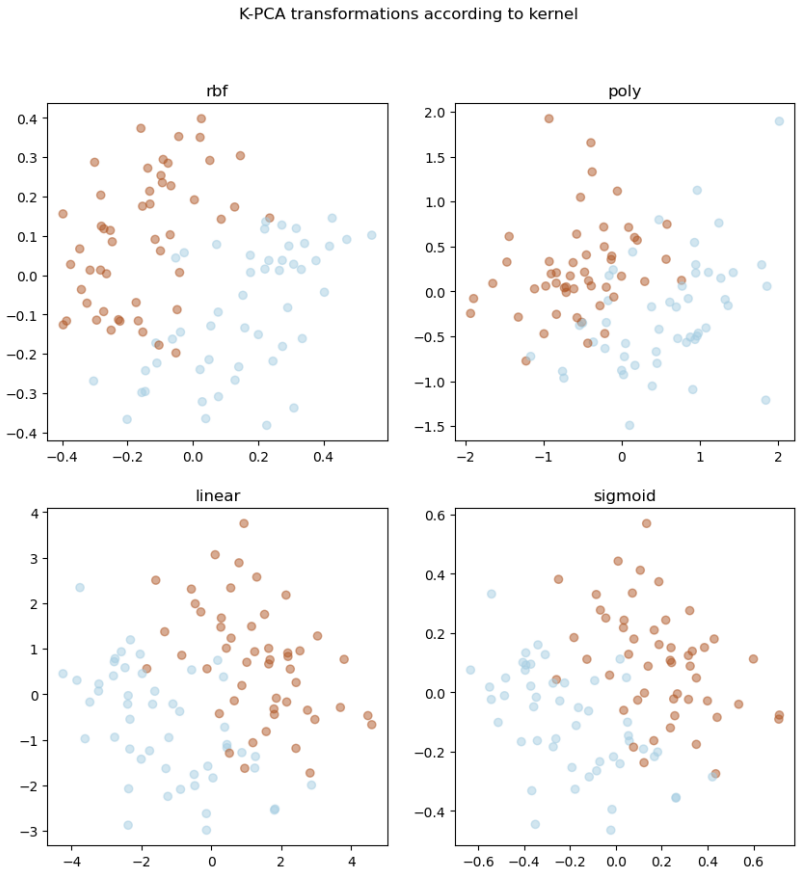

Figure 9: KPCA Transformations in 2D

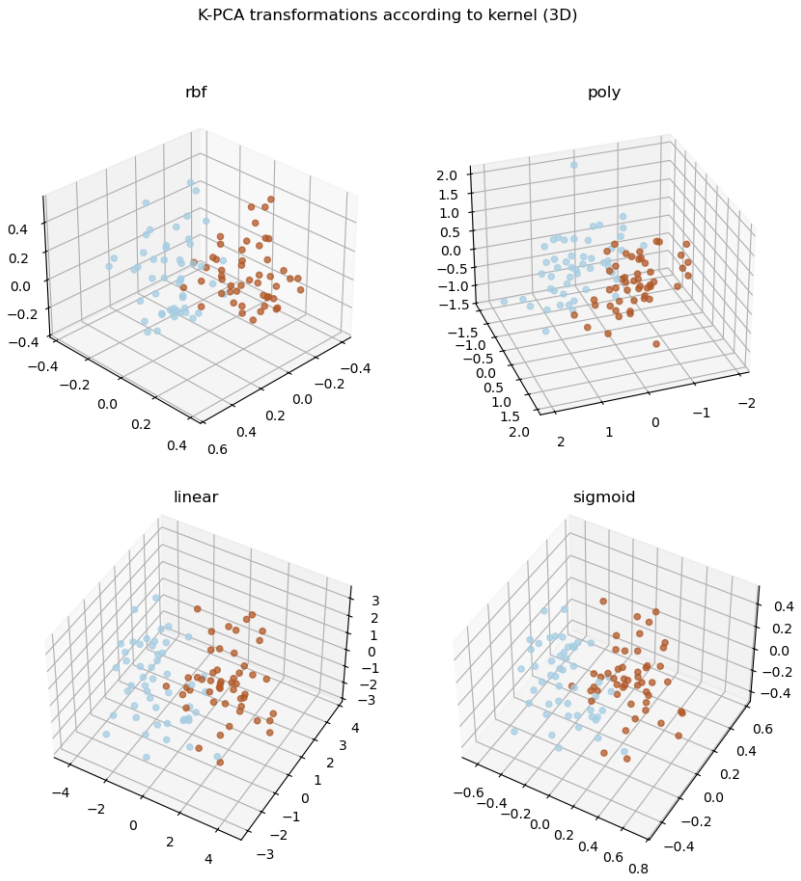

Figure 10: KPCA Transformations in 3D

Discussione

Dataset 1

Osservando le performance dei modelli, è evidente che la kernelizzazione è un fattore necessario per affrontare il problema.

Infatti, il modello non-kernel (ossia la Ridge Regression) riporta un punteggio R2 basso (0.2263, tbl. 3) e dalla rappresentazione grafica del dataset si evince che il modello non è abbastanza complesso per spiegare la non-linearità della funzione target (fig. 1, eq. 1).

Dataset 2

Dataset 3

Conclusione

SCALETTA: - Introduzione: Obbiettivi (kernelizzazione vs non kernelizzazione) e quali dataset useremo (generati artificialmente) - Testare la regressione ridge non kernelizzata vs kernelizzata, con un dataset composto da funzioni trigonometriche e lineari (quindi a "gobbe") - Testare metodi di riduzione della dimensione kernelizzate (PCA), col make_circles dataset e con dataset più complessi (make_classification()) - Metodologia: descrivo COSA ho fatto, suddivido in tre parti come nel notebook 1 - EDA: vedo com'è fatto il dataset, per farmi un'idea - Effettuo e valuto una ridge regression - Eseguo due grid search, una per il KRR gaussiano, l'altro per il KRR polinomiale - Plotto le funzioni che variano a seconda dei parametri - Valuto i loro modelli migliori

```
2
- EDA: Visualizzo, effettuo PCA e proietto sia in due che una dimensione.
- Fitto una SVM lineare e lo valuto usando il test di testing
- Effettuo PCA kernelizzata (con gauss), proietto sia in 2D che 1D
- Fitto SVM kernelizzata e valuto

3
- Effettuo PCA in due e tre dimensioni
- Fitto SVM lineare e valuto
- Effettuo K-PCA con tutti i kernel possibili e proietto in due e tre dimensioni
- Per determinare quale kernel usare e quali iperparametri mettere, effettuo l'hyperparameter tuning con Ha
```

- Risultati: riporto grafici e performance
- Discussione: suddivido in tre parti

  1. Chiaramente il modello lineare fallisce di descrivere bene la funzione target (vedere figura!) I modelli kernelizzati sono abbastanza sensibili ai loro parametri:
     - Gaussiano: gamma determine quanto il modello vada a "underfittare" o "overfittare" sul problema. In effetti determina l'ampiezza \sigma.
     - Polinomiale: degree determina la complessità del fitting (nel nostro caso più alto, meglio era) e alpha determina la "smoothness" della funzione imparata
  2. la PCA è insufficiente per ridurre la dimensionalità del dataset, infatti non è linearmente correlata. Usando il kernel riesco a creare una proiezione (sia in 2D che 1D) dove le classi sono linearmente separabili.

  Le applicazioni del modello SVM e kernel SVM confermano

  3. Il dataset è più complesso, in effetti ha dieci variabili

  Ciò ha reso il processo della selezione del kernel e i suoi parametri più difficile, infatti abbiamo usato metodi di ottimizzazione degli iperparametri più sofisticati (halving random search CV)

- Conclusione: riassumo gli effetti della kernelizzazione e meno sui dataset
  - La kernelizzazione tende a migliorare i risultati, specie quando il dataset è complesso

– Tuttavia notiamo che è “sensibile” ai parametri che scegliamo