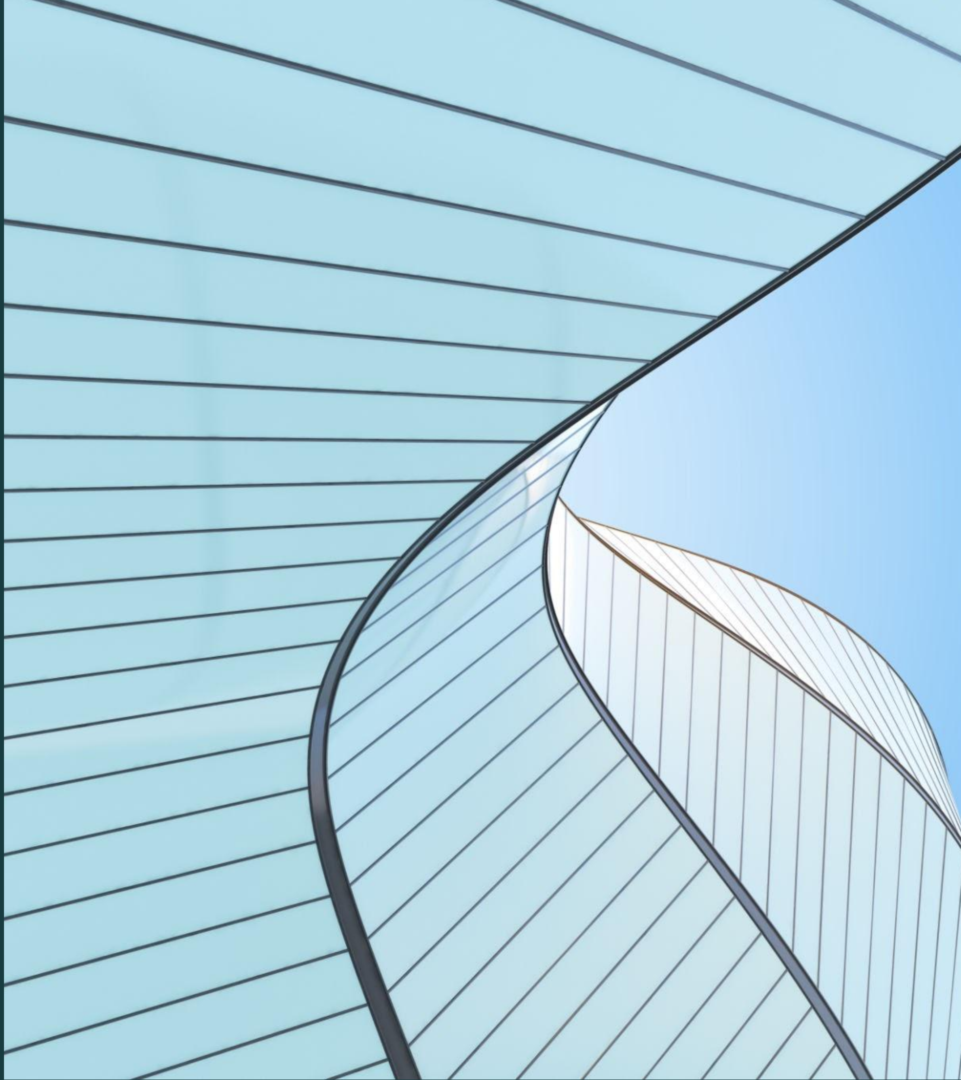
Mohammed Aldahmani
ML702 Spring 2025
April 28th 2025

DC-KRR for Large-Scale Learning

Million Song Dataset Project



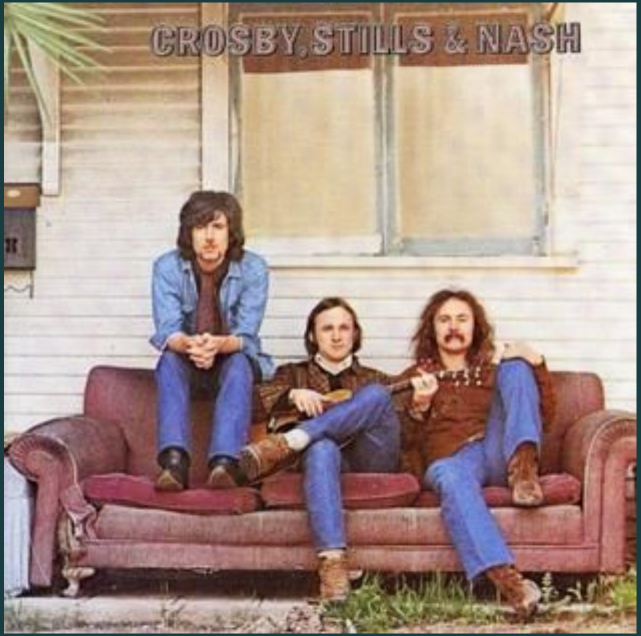
Introduction

Kernel methods enable flexible nonlinear modeling.

Kernel Ridge Regression (KRR) adds l2 regularization.

Large datasets make direct KRR infeasible.

Goal: Apply KRR to the Million Song Dataset 500K rows 90 columns



Literature Review

Theodoros Evgeniou et al. (2000)

This paper formalizes Kernel Ridge Regression (KRR) by extending Ridge Regression into nonlinear spaces via kernels. It introduces the representer theorem and connects regularization to kernel-based learning.

Rifkin and Lippert (2007)

Ridge Regression stabilizes linear models through l2 regularization. The paper emphasizes cross-validation for selecting the regularization parameter lambda and shows that Ridge has efficient closed-form solutions.

Zhang, Duchi, Wainwright (2015)

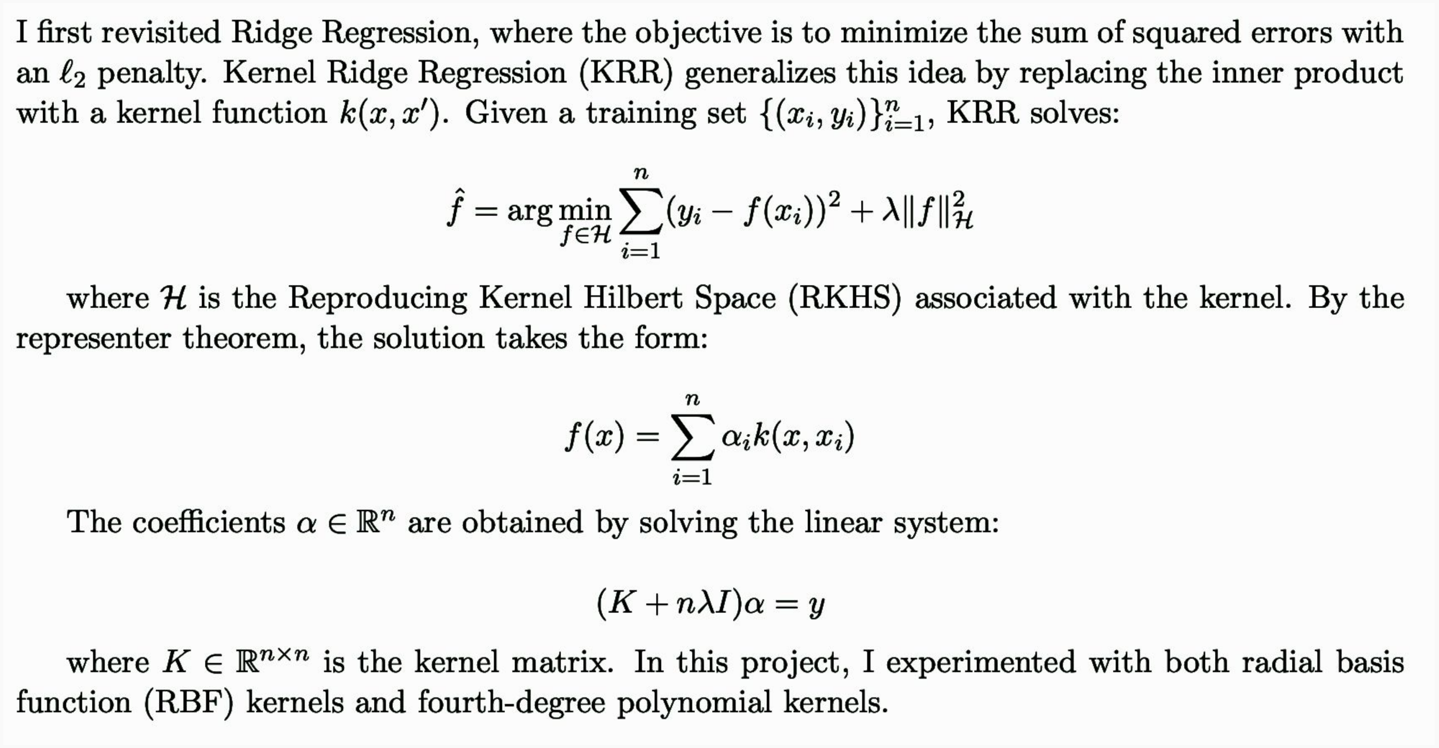
They propose a divide-and-conquer strategy for KRR to address memory and computational challenges. Training on small subsets and averaging predictions achieves scalable learning with strong theoretical guarantees.

Introduce a
ess obj

To establish hov
easure succes
perfor
(KPIs)
this:

→ KPI 1
→ KPI 2
→ KPI 3
→ KPI 4

Theoretical Derivation



I first revisited Ridge Regression, where the objective is to minimize the sum of squared errors with an ℓ_2 penalty. Kernel Ridge Regression (KRR) generalizes this idea by replacing the inner product with a kernel function $k(x, x')$. Given a training set $\{(x_i, y_i)\}_{i=1}^n$, KRR solves:

$$\hat{f} = \arg\min_{f \in \mathcal{H}} \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \|f\|_{\mathcal{H}}^2$$

where $\mathcal{H}$ is the Reproducing Kernel Hilbert Space (RKHS) associated with the kernel. By the representer theorem, the solution takes the form:

$$f(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i)$$

The coefficients $\alpha \in \mathbb{R}^n$ are obtained by solving the linear system:

$$(K + n\lambda I)\alpha = y$$

where $K \in \mathbb{R}^{n \times n}$ is the kernel matrix. In this project, I experimented with both radial basis function (RBF) kernels and fourth-degree polynomial kernels.

Methodology: Implementation Details

Full KRR infeasible:

- $O(n^2)$ memory (roughly ~64GB needed of RAM).
- $O(n^3)$ time (too slow).

Running on my Local machine

Solution: Divide-and-Conquer KRR (DC-KRR) with 100/50/40 subsets.

Baseline Ridge Regression ($O(nd)$ complexity) trained efficiently.

Naive KRR trained on 10k subsample for comparison.

Dataset:Million Song Dataset (Kaggle).

460,000+ examples, 90 features.

Target: Year of song release.

80% train / 20% test split.

Experimental Setup

Baseline Ridge Regression:

- Trained on full data.

RBF KRR:

- Failed (MSE in tens of millions).

Polynomial KRR (degree 2):

- Improved stability.

Naive subsample KRR (10k points).

DC-KRR:

- 100/50/40 disjoint subsets.

Results

Ridges Regression:
MSE approx 93

Naive Subsample (10k points):
MSE approx 22,912.

DC-KRR (m=44 subsets):
MSE approx 6,174.

DC-KRR (m=30 subsets):
MSE approx 4,009.

Divide-and-Conquer significantly improved stability and error.

Discussion

- **Direct KRR infeasible on local machine.**
- **RBF kernels performed poorly on this dataset.**
- **Polynomial kernels stabilized learning.**
- **Divide-and-Conquer strategy critical for scaling kernel methods.**
- **Limited time for extensive tuning, but clear insights gained.**

Conclusion

Divide-and-Conquer made KRR feasible on large datasets.

Achieved major MSE reduction: 40M to 4K.

Highlights importance of scalability and kernel selection.

Future work: deeper tuning, more feature engineering and alternative kernels.

Thank you

Questions?