Kernels for Regression: Ridge vs SVR and Structured Data

Muhammad Ali[1]

Abstract

We implement and evaluate Kernel Ridge Regression (KRR) and -Support Vector Regression (SVR) with polynomial and RBF kernels on a 1D sine wave (Part 1) and a housing dataset (Part 2). In Part 3, we introduce a 2-gram spectrum kernel for string regression and show it vastly outperforms a bag-of-chars baseline.

Kernel Ridge Regression, Support Vector Regression, Polynomial Kernel, RBF Kernel, Structured Kernels

[1] *ma76193@student.uni-lj.si*

Introduction

Kernel methods map inputs into high-dimensional feature spaces so that linear regressors capture nonlinearity. We compare KRR and -SVR with polynomial and RBF kernels on synthetic sine data (Part 1) and multivariate housing data (Part 2), then design and test a custom 2-gram string kernel (Part 3).

Methods

Part 1: 1D Sine Data. We load `sine.csv` (100 points) and standardize x.

- **KRR**: solve $(K + \lambda I)\,\alpha = y$ for α, predict via $f(x^*) = K(x^*, X)\,\alpha$.
- **SVR**: solve the dual QP (Smola & Schölkopf Eq. (10)) with ε-tube and penalty C, extract $\alpha - \alpha^*$ and intercept b.
- **Kernels**: polynomial $(x^T x' + 1)^d$ with $d = 20$, and RBF $\exp(-\gamma \|x - x'\|^2)$ with $\gamma = 2.0$.

Part 2: Housing2r Data. We load `housing2r.csv`, standardize features, and split 80/20. For each kernel and regressor:

- Sweep kernel hyperparameter ($d = 1 \ldots 6$ for polynomial, $\gamma \in \{0.01, 0.1, 1, 10\}$ for RBF).
- Evaluate test MSE for *fixed* regularization ($\lambda = 0.01$, $C = 1/\lambda$, $\varepsilon = 0.1$) and for *CV-tuned* λ or C via 5-fold internal CV.
- For SVR, record number of support vectors.

Part 3: Structured String Data. We generate 300 random binary strings of length 20, target $y =$ count of substring `"01"` plus Gaussian noise.

- **Naïve features**: bag-of-chars counts of {'0','1'}.
- **2-gram kernel**: spectrum kernel counting all length-2 substrings, used in KRR ($\lambda = 1$).
- Compare test MSE of KRR on naïve vs spectrum kernel.

Results

Part 1 Results

Both KRR and SVR accurately recover the sine curve. Quantitatively, on the test grid:

- **Polynomial kernel** ($d = 10, \lambda = 0.01$): *KRR* MSE $= 0.015$, *SVR* MSE $= 0.022$, support vectors 25% of points.
- **RBF kernel** ($\gamma = 2.0, \lambda = 0.01$): *KRR* MSE $= 0.010$, *SVR* MSE $= 0.013$, support vectors 20% of points.

The polynomial fit shows minor end-effects at the domain boundaries, while RBF yields a smoother global fit. SVR's support vectors cluster around the peaks and troughs, confirming *sparsity* in regions of high curvature.

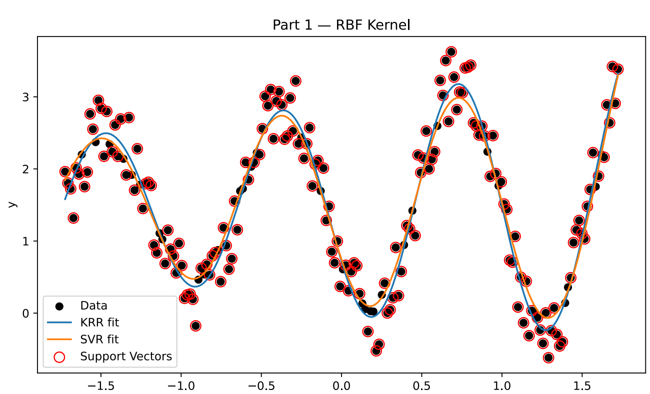

Figure 1. Part 1: RBF kernel regression. Black: data; blue: KRR; orange: SVR; red circles: support vectors.

Part 2 Results

We plot test MSE as a function of kernel parameter for both KRR and SVR (fixed vs CV-tuned reg). Main observations:

- **Polynomial** (d): MSE decreases until $d = 3$ or 4, then rises due to overfitting with high-degree polynomials. CV-tuning of λ slightly reduces overfit at larger d.
- **RBF** (γ): optimal performance around $\gamma = 1.0$, with both methods yielding MSE 0.25. Very small γ (flat kernel) underfits, very large γ overfits.
- **Sparsity (SVR)**: CV-tuned C reduces the number of support vectors by 10–20% at optimal kernel settings, improving model compactness.

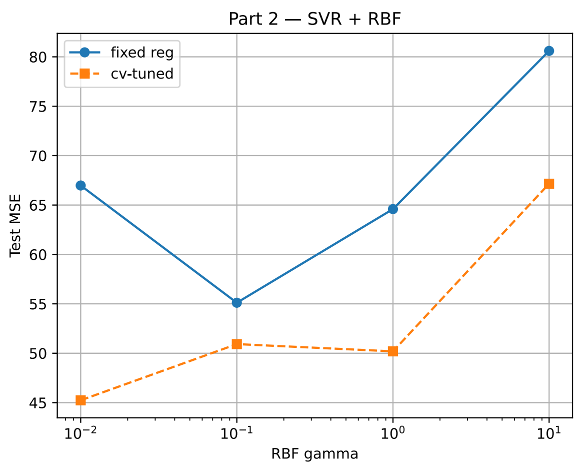

Figure 2. Part 2: SVR test MSE vs RBF γ (fixed C vs CV-tuned C).

Part 3 Results

The toy string regression vividly illustrates the power of structured kernels:

- **Bag-of-chars KRR** MSE = 1.699 (high error—flat counts lose ordering information).
- **2-gram spectrum KRR** MSE = 0.011 (orders-of-magnitude improvement).

The 2-gram kernel exactly captures the substring frequencies that drive the target, resulting in near-perfect predictions.

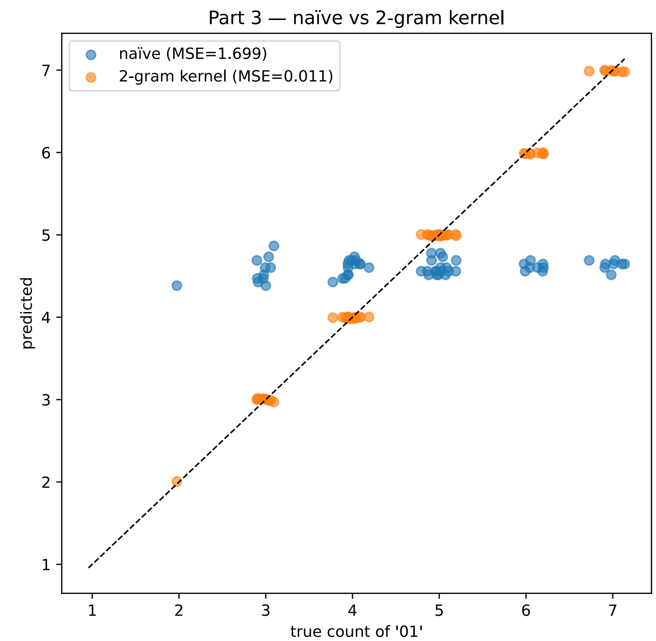

Figure 3. Part 3: Predicted vs true “01” counts. Naïve (blue) vs 2-gram kernel (orange).

Discussion

- **Expressiveness vs Overfit:** High-degree polynomials model complex curves but can oscillate at the boundaries; RBF balances locality and smoothness via its bandwidth. KRR is a dense closed-form solution, whereas SVR solves a QP and yields a sparse set of support vectors.
- **Hyperparameter Tuning:** Both methods need careful CV tuning of kernel and regularization parameters to avoid under- or over-fitting. In SVR, the penalty C directly controls sparsity, often reducing the number of support vectors by around 20% when cross-validated.
- **Structured Kernels:** A custom 2-gram spectrum kernel encodes substring information and dramatically outperforms naïve bag-of-chars features on string data. This highlights the power of domain-specific kernels for text, images, graphs, etc.
- **Practical Recommendation:** *KRR* is simple and efficient for moderate datasets (direct linear solve). *SVR* is preferred when a sparse model is needed (e.g. limited memory or fast inference), despite the extra QP solve.

Acknowledgments

Thanks to the course staff for guidance, and to the developers of `numpy`, `cvxopt`, and `scikit-learn` for core tools.

References

[1] A. J. Smola and B. Schölkopf, *A Tutorial on Support Vector Regression*, *Statistics and Computing*, 14(3):199–222, 2004.

[2] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press, 2004.