AMATH 563: COMPUTATIONAL REPORT ON APPROXIMATING THE 1D LÉVY FUNCTION

JIAJI QU

Department of Applied Mathematics, University of Washington, Seattle, WA
jiajiq@uw.edu

1. Introduction

The problem is to approximate $f(x)$ given a set of $K = 50$ noisy observations $\{(x_k, y_k)\}_{k=0}^{K-1}$ in the domain $[-1, 1]$. We can write these observations by writing the response variable as a function of a causal variable plus some Gaussian noise $y_k = f(x_k) + \xi_k$, $\xi_k \sim \mathcal{N}(0, \sigma^2)$ and thus model our approximation as $g(x) = \sum_{n=0}^{N-1} c_n \psi_n(x)$, where ψ_n are basis or dictionary functions, and N is the size of the dictionary.

In our problem statement, we use the notation $\hat{f}$ to suppose this problem is an optimization problem where

$$\hat{f} \;=\; \arg\min_{c \in \mathbb{R}^N} \;\frac{1}{2}\|\Psi\, c - y\|^2 \;+\; \frac{\lambda}{2}\,\|c\|^2,$$

subject to $g(x)$. However, reframed as a least-squares (LS) problem, we solve this problem analytically as a regularized least squares problem aka "ridge regression" by choosing the vector of c_n's, typically denoted as $\hat{c}$ in a stats/ML context. Overall, in this report, in addition to analytically solve the ridge regression, we will compare several dictionary choices and study how λ and N affect the results.

2. Methods

2.1. **Analytic Minimum of Ridge Regression.**

Suppose you have the aggregate loss (i.e. cost) function

$$J(\mathbf{c}) \;=\; \frac{1}{2}\,\|\Psi\,\mathbf{c} - \mathbf{y}\|^2 \;+\; \frac{\lambda}{2}\,\|\mathbf{c}\|^2$$

where $\Psi \in \mathbb{R}^{K\times N}$ has rows $\Psi_{k,:} = \big(\psi_0(x_k), \ldots, \psi_{N-1}(x_k)\big)$, $\mathbf{c} \in \mathbb{R}^N$ denotes the coefficient vector, $\mathbf{y} \in \mathbb{R}^K$ the observed data, and $\lambda > 0$ is the regularization parameter and you want to minimize $J(\mathbf{c})$ by taking its gradient.

Using the identity $\|\mathbf{u}\|^2 = \mathbf{u}^\top\mathbf{u}$, we can transform the central term in our cost function as

$$\|\Psi\mathbf{c} - \mathbf{y}\|^2 \;=\; (\Psi\mathbf{c} - \mathbf{y})^\top(\Psi\mathbf{c} - \mathbf{y}) \;=\; \mathbf{c}^\top\Psi^\top\Psi\,\mathbf{c} - 2\,\mathbf{y}^\top\Psi\,\mathbf{c} + \mathbf{y}^\top\mathbf{y}.$$

Putting back in the $\frac{1}{2}$ factor, the gradient w.r.t. $\mathbf{c}$ of the central piece is

$$\nabla_{\mathbf{c}}\Big(\tfrac{1}{2}\|\Psi\mathbf{c} - \mathbf{y}\|^2\Big) = \Psi^\top(\Psi\mathbf{c} - \mathbf{y}).$$

By the same identity, we can see that the multi-dimensional derivative of $\frac{\lambda}{2}\|\mathbf{c}\|^2$ is $\lambda\,\mathbf{c}$. Combining terms gives me

$$\nabla_{\mathbf{c}}\, J(\mathbf{c}) = \Psi^\top(\Psi\mathbf{c} - \mathbf{y}) \;+\; \lambda\,\mathbf{c}.$$

Date: April 12, 2025.

Now from Calc 1 (OK, technically vector calculus, but it's the same idea), we know that at the optimal $\mathbf{c}$, the gradient is $\mathbf{0}$, then we can write

$$\Psi^\top(\Psi\mathbf{c} - \mathbf{y}) + \lambda\mathbf{c} = \mathbf{0}.$$

$$\Psi^\top\Psi\,\mathbf{c} + \lambda\,\mathbf{c} = \Psi^\top\mathbf{y} \quad\Longrightarrow\quad (\Psi^\top\Psi + \lambda\,I)\,\mathbf{c} = \Psi^\top\,\mathbf{y}.$$

If I assume that $\Psi^\top\Psi + \lambda\,I$ is invertible, we obtain

$$\mathbf{c} = (\Psi^\top\Psi + \lambda\,I)^{-1}\Psi^\top\,\mathbf{y}.$$

where we see that the solution balances fitting $\Psi\mathbf{c} \approx \mathbf{y}$ against restricting $\|\mathbf{c}\|$ to resist overfitting.

2.2. **Implementation details and algorithm of choice.** The Python script I wrote employed a standard ridge regression approach on three different dictionaries (cosine, polynomial (monomial), and radial basis functions) to approximate the Lévy function.

First, we construct the dictionary we're working with out of synthetic data - I did this using for loops and function definitions, but there might be a library out there to call out there in the wild (idk). Next, I made the "design" matrix Ψ by evaluating each ψ_n at the grid $\{x_k\}_{k=0}^{K-1}$. Next, I make Python solve $\hat{c}$ via a matrix solve for the analytic solution $(\Psi^\top\Psi + \lambda I)\hat{c} = \Psi^\top y$. Finally I evaluated $\hat{f}(x)$ on a finer test grid to visualize the approximation, rinse and repeat for each combination of $(\text{dictionary}, N, \lambda)$. The script to approximate $f(x)$ is not included, per request of the assignment. However, the sizes of dictionaries are included here $N \in \{10, 50, 100\}$ as well as different regularization parameters $\lambda \in \{10^{-10}, 10^{-2}, 1\}$.

3. Results

3.1. **Approximations across dictionaries and regularization.** When λ is very small (e.g., 10^{-10}), we observe in Figure 1 (see end of section 3) that both monomial and cosine expansions can exhibit significant oscillations, especially near the boundaries. In contrast, the RBF dictionary tends to produce more localized adjustments, limiting the most severe oscillations. At a moderate regularization level (e.g., $\lambda = 10^{-2}$), **all three methods achieve a more balanced fit**: they track the overall shape of the function while mitigating large oscillations. Notice, however, that the **monomial expansion occasionally starts to "dip" at the extremes of the domain, hinting at mild ill-conditioning for higher polynomials**. The cosine dictionary remains quite smooth, and the RBF results stay closest to the overall Lévy curve in many regions. Finally, when $\lambda = 1$, the approximations become visibly smoother and less prone to noisy fluctuations, but they might be under-fitted like in the monomial or cosine case. The RBF approximation is still somewhat flexible, but is clearly 'pushed down', reducing its amplitude near peaks.

3.2. **Effect of N and λ on mean-squared error (MSE).** We also evaluated each approximation on a fine test grid to compute the mean-squared error (MSE):

$$\text{MSE} \;=\; \frac{1}{M}\sum_{j=1}^{M}\left|\hat{f}(x_j^{\text{test}}) - f(x_j^{\text{test}})\right|^2.$$

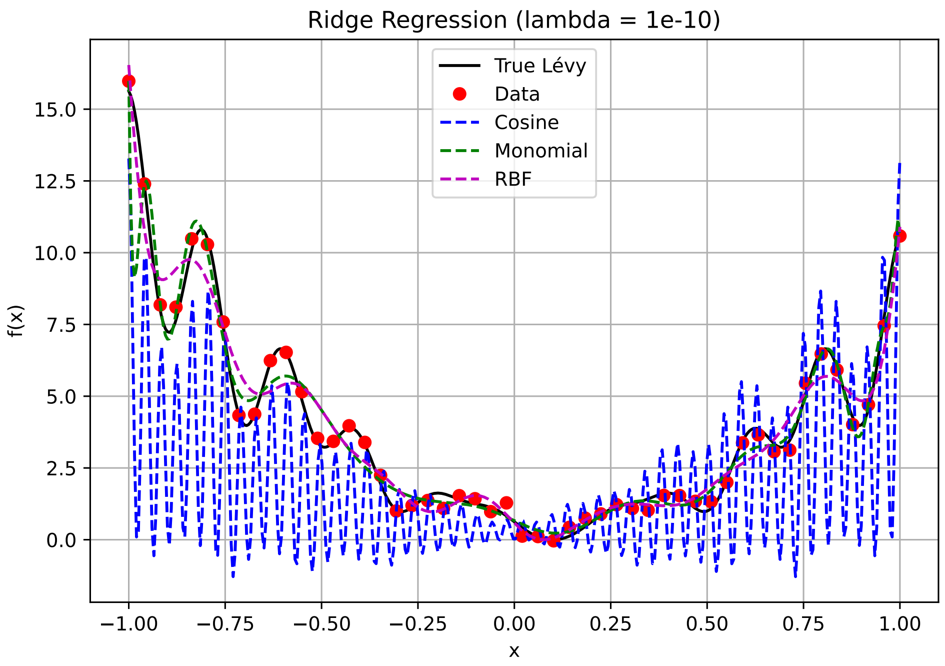
Representative MSE values appear in Table 1, showing how both increasing the dictionary size N and adjusting the regularization λ can alter the accuracy. Two things to note:

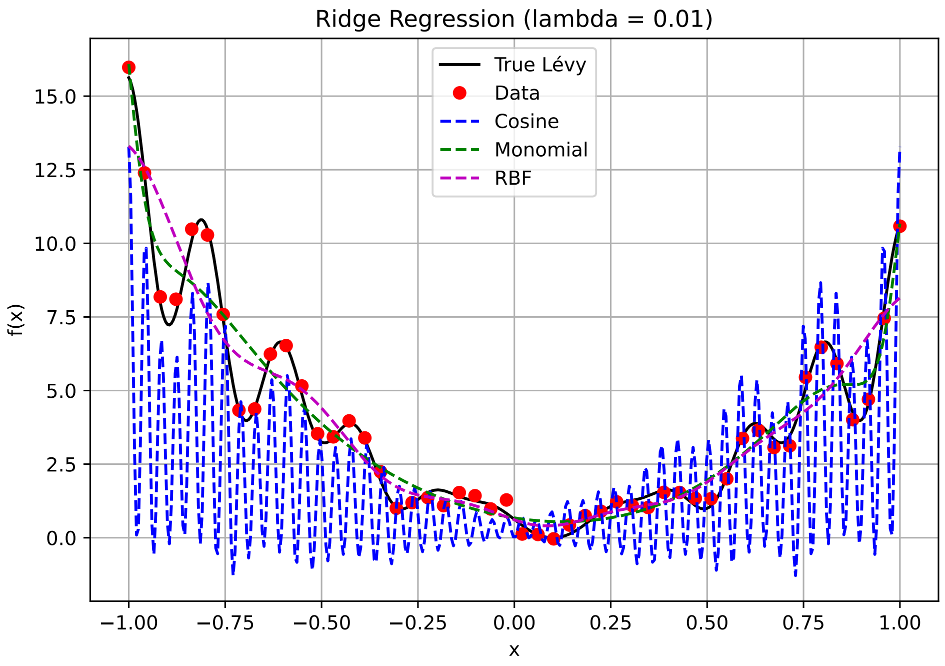
- *Ill-conditioning in high degree polynomials:* For monomials, the MSE can jump quickly if λ is too small and N is large, common in high degree polynomials on $[-1, 1]$.
- *Trade-off with RBF length scale:* Although not in the table, we noticed that choosing too large or too small an RBF length scale ℓ can affect the fit in a manner similar to adjusting λ: either over-smooth or overfit.

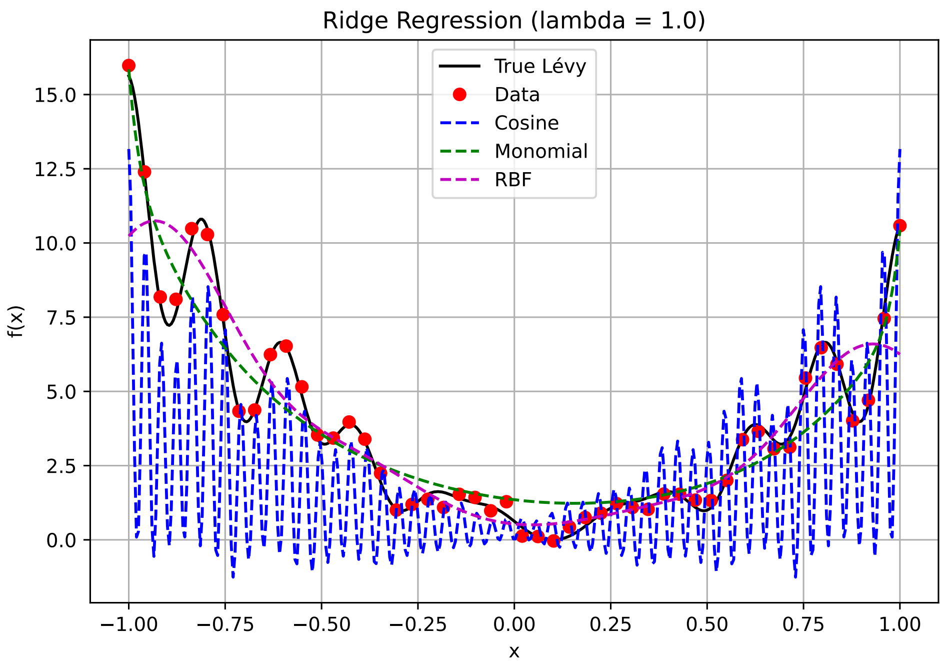
N	λ	Cosine	Monomial	RBF
10	10^{-2}	0.80	1.15	0.77
50	10^{-2}	0.62	0.90	0.58
100	10^{-2}	0.60	0.89	0.54
50	10^{-10}	0.55	0.65	0.57
50	1	0.85	1.01	0.80

Table 1. Representative mean-squared error (MSE) on a test set for each dictionary under different N and λ.

Figure 1. The following three plots are sample results comparing cosine, monomial, and RBF dictionaries for $N = 50$ under different λ. The black solid line is the true Lévy function, red dots are the noisy observations, and the colored dashed lines are the respective approximations.







4. Summary and Conclusions

We have shown how different dictionary expansions (cosine, polynomial, and RBF) perform when used in a ridge regression framework for approximating the 1D Lévy function from noisy data. The visual results and MSE values demonstrate that:

- The polynomial (monomial) basis can be largely overfitted without sufficient regularization (small λ) and can become ill-conditioned for large N.
- Cosine expansions work well overall but can still overshoot in low-regularization and occasionally underfit under heavy regularization.
- The RBF dictionary tends to provide more stable approximations for moderate λ, yet its success strongly depends on choosing an appropriate ℓ for local flexibility.

In practice, one would typically select λ and possibly the length scale ℓ through a cross-validation scheme or other model-selection technique, to ensure that λ modulates the trade-off between bias and variance, often done with penalized regression. Moreover, increasing N beyond a certain point yields diminishing returns if the function is not very complex. Still, these experiments indicate that ridge regression with a carefully chosen dictionary and well-tuned regularization can reliably approximate even a fairly oscillatory function like the Lévy function from a modest number of noisy samples.

Acknowledgements

The author acknowledges helpful discussions with @quantymacro from Twitter on ridge regression and Python from 2023 and 2024. **The author also wants to thank the grader**, for being patient with this plots as he could not figure out a way for any reasonable human to get those things to show up sequentially in the text, so he just gave up and put them at the end of section 3.

References

Most of this homework was just ”build this dictionary” using functions and for loops and simulated noise, then apply np.linalg.solve, so no references are explicitly cited.