Machine Learning - Homework 5

Tim Kmecl, 8.5.2024

Implementation

I first implemented the two kernels without using any for loops, just using matrix operations and numpy functions. I compared the results to kernels imported from sklearn and they match.

Kernelized ridge regression was implemented based on the equation from the lecture video. SVR is implemented by using cvxopt.solvers.qp to solve the optimization problem. I also wrote a function that identifies support vectors based on alphas, with a tolerance of 1e-8. All the tests for both methods pass.

Sine

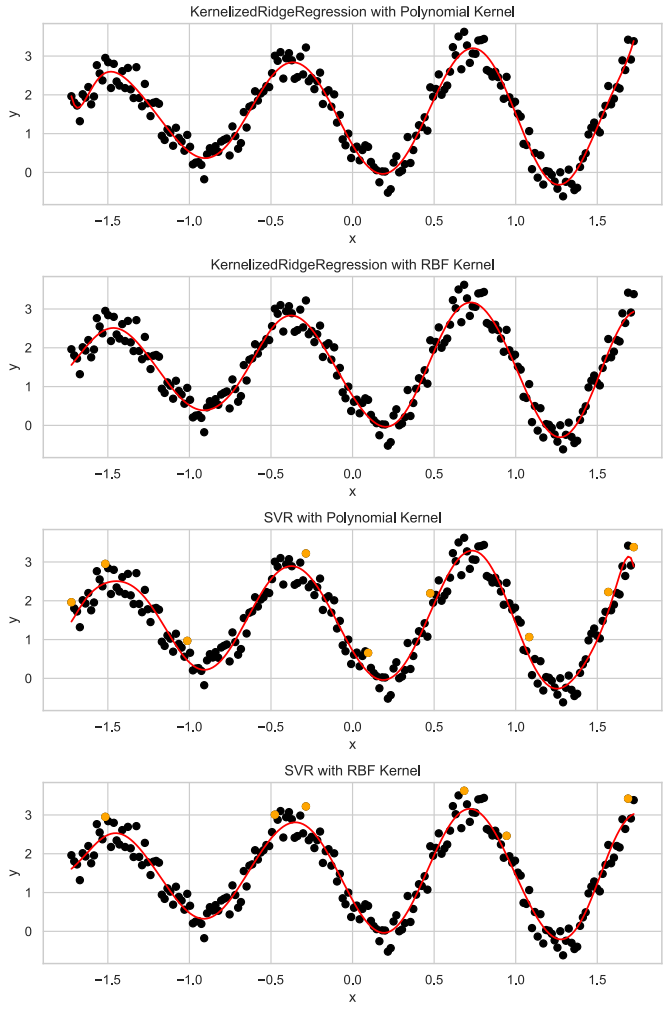

Figure 1: Results of fitting to the sine data. Support vectors are orange dots.

I applied both methods to the Sine dataset, where the input variable was standardized beforehand. For both methods, I tried both kernels with different kernel parameters and different regularization parameters, and have also experimented using different values of epsilon for SVR.

The larger the epsilon, the less support vectors there are. Too large epsilon causes the fit to become worse due to too few support vectors. When using large M or small sigma with too small regularization parameter, overfitting becomes visible due to larger flexibility of the model – this can be countered by increasing lambda. Data, fits and support vectors are shown in figure 1.

For KRR with Polynomial kernel, final parameters are M=15 and λ=0.01. For KRR with RBF kernel, σ=0.2 and λ=0.2. For SVR with Polynomial kernel, M=15, λ=0.0001 and ε=0.5 which results in 9 support vectors. For SVR with RBF kernel, σ=0.2, λ=0.1 and ε=0.5, which results in 6 support vectors. The MSE for all 4 approaches is around 0.08.

Housing

First I fitted both models with both fitters with different parameters on the first 80% of the dataset and evaluated the performance on the last 20%.

I tried 10 different values for M (1,2,3, ...10) and for sigma (0.2, 0.4, 0.6, ... 2). I set epsilon to 10 to limit the number of support vectors. The results are shown in figure 2 in blue.

I repeated the experiment using internal 4-fold CV to choose lambda. Lambdas that were tried were from the logarithmic space, from 10^-5 to 10^5 insteps in exponents of 0.5 for KRR and 1 for SVR. The best lambda for each configuration was then evaluated on the test set. The results are shown in figure 2 in red. (epsilon was selected by hand based on performance and scarcity of support vectors, I picked 7).

Both models are similar in that we have to take certain measures to prevent overfitting (at small sigmas and especially large Ms). In SVR we can also help this not just with lambda, but by increasing epsilon, which decreases the amount

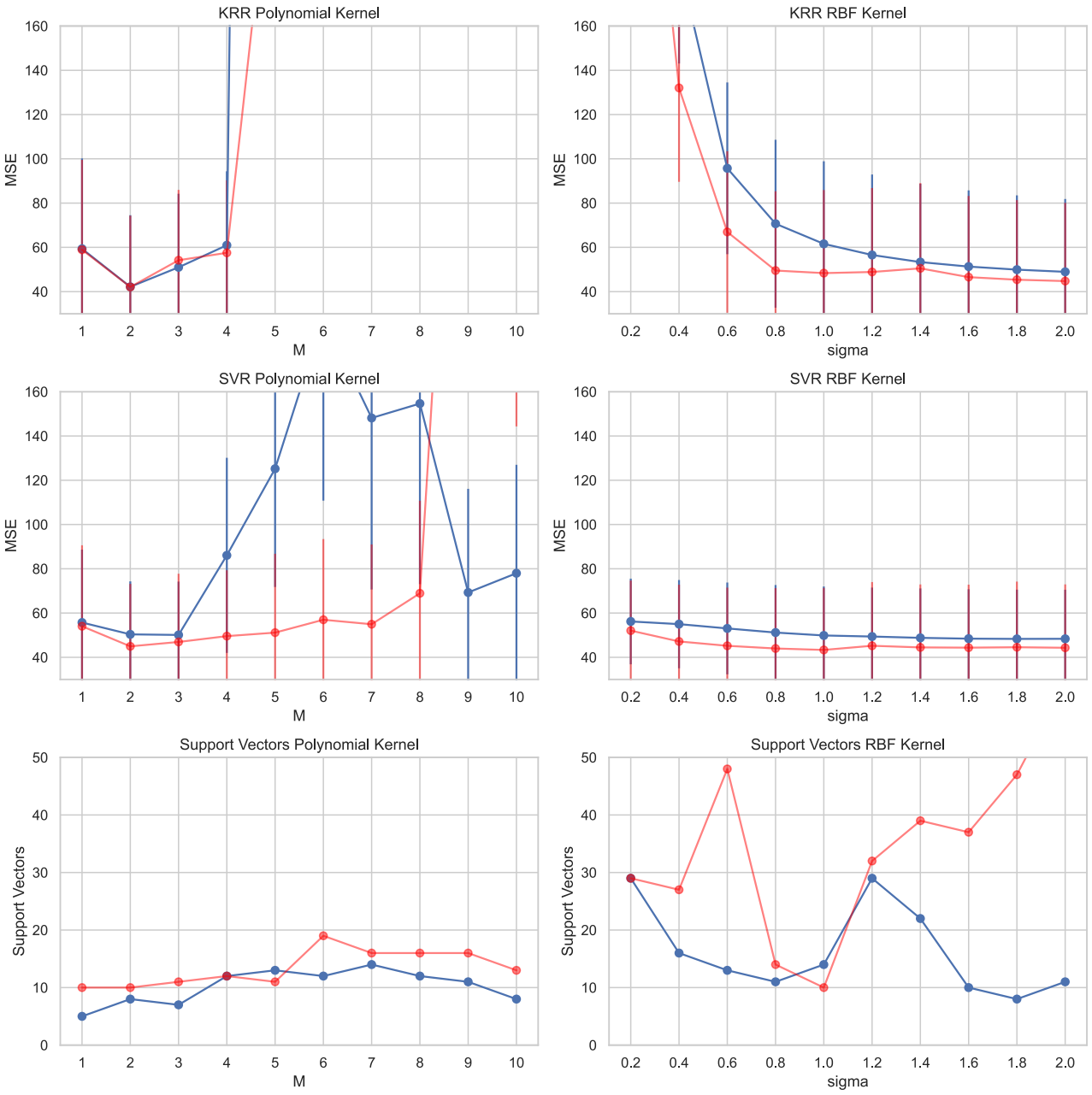

Figure 2: Results on the Housing dataset. Blue lines are for fixed lambda=1, red for lambdas chosen using internal CV.

of support vectors and makes the model less flexible, preventing overfitting.

As we see from the plots for Polynomial kernel, MSEs increase after some point in both cases, both with lambdas chosen with CV and when fixed, however it is clear that SVR is more robust, with MSEs increasing way slower. Despite that, there is still a spike at M=9 for lambdas chosen using CV (worse performance compared to fixed lambda is probably because of lower epsilon).

KRR with RBF kernel performs badly at smaller sigmas, with selecting lambda benefiting the performance. SVR on the other hand performs well at all sigmas, but again performs better at higher ones (the difference here is a lot smaller). Selecting lambdas using CV again helps.

In light of all this, I would prefer SVR due to it being more robust, with performance remaining more stable no matter the choice of kernel parameters. It also gives us more flexibility with both lambda and epsilon to pick. Moreover, support vectors also give us additional interpretability options.