Machine Learning for Data Science HW4 SVM

Nikolay Kormushev

Introduction

In this report, I implement a kernalized ridge regression and support vector regression using the cvxopt library. The goal of this report is to get a deeper understanding of the two methods and compare them to determine when each performs better.

Implementation

Kernalized ridge regression

The implementation of kernalized ridge regression is pretty straightforward. I implement the following formula to make my predictions:

$$y(x') = k(x')(K + \lambda I_n)^{-1} y_{train} \tag{1}$$

where x' is the sample I want to predict λ is the regularization parameter of ridge regression and I_n is the nxn identity matrix. K and k are given bu the following formulas where κ is the kernel function.

$$k(x') = \begin{bmatrix} \kappa(x', x_1) \\ \vdots \\ \kappa(x', x_n) \end{bmatrix} \text{ and } K = \begin{bmatrix} \kappa(x_1, x_1) & \cdots & \kappa(x_1, x_n) \\ \vdots & \ddots & \vdots \\ \kappa(x_n, x_1) & \cdots & \kappa(x_n, x_n) \end{bmatrix}$$

Support Vector Regression

To implement support vector regression I solved the optimization problem given in equation 10 in [1] using cvxopt. For that I turned the maximization problem into a minimization one by adding a minus to the equation. After I had to calculate the parameters required for the cvxopt optimizer. Parameters were considered in the following order $[\alpha_1, \alpha_1^*, \alpha_2, \alpha_2^*, \alpha_3, \alpha_3^*, \ldots]$ My results are the following:

$$G = \begin{bmatrix} -1 & 0 & \cdots & 0 \\ 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & -1 \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

$$A = [1, -1, 1, -1, \ldots], h = [0, 0, \ldots, 0, C, C, \ldots, C]$$
$$q = [\varepsilon - y, \varepsilon + y, \varepsilon - y, \varepsilon + y, \ldots, \varepsilon - y, \varepsilon + y], b = 0$$

$$P = \begin{bmatrix} k(x_1, x_1) & -k(x_1, x_1) & \cdots & k(x_1, x_m) & -k(x_1, x_m) \\ -k(x_1, x_1) & k(x_1, x_1) & \cdots & -k(x_1, x_m) & k(x_1, x_m) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ k(x_m, x_1) & -k(x_m, x_1) & \cdots & k(x_m, x_m) & -k(x_m, x_m) \\ -k(x_m, x_1) & k(x_m, x_1) & \cdots & -k(x_m, x_m) & k(x_m, x_m) \end{bmatrix}$$

Once I had the parameters I input them into the optimizer and which gives me the alphas and I calculate the weights $w_i = \alpha_i - \alpha_i^*$. To predict I use this formula: $y(x') = w^T \cdot \kappa(X_{\text{train}}, x') + b$

Testing

For testing I used the sine dataset and compared my predictions to the ones from scikitlearns implementation of SVR and kernal ridge. I computed the MSE between the predictions and compared and saw it was extremely small(around 1e-25) which proves my methods work.

Results

Evalution

For evaluation in both cases I do an 80/20 train test split. I standardize the data and for the housing I do cross validation to find my results.

Sine

In Figure 1 I have show the resulting hyperplanes from using the ridge and support vector regressions with the two different kernels. I have also plotted the training and testing data(the blue and magenta dots) and the support vectors. The support vectors are in different colors based on the kernel. The solutions shown might not be the optimal ones. I picked them manually so we can compare the methods.

From the plot we can see that all models manage to approximate the sine wave quite well which normally wouldn't be possible for a linear regression. This shows that transforming the data does have an effect.

Parameter influence

For the polynomial kernel I needed a high degree to approximate the data which is to be expected due to the large amount of curves.

For the sigma I chose a small value which means only the closer points would effect our predictions which introduced more variance and less bias as opposed to bigger sigmas which did not learn much and gave straight lines. Small sigmas on the other hand did not have enough points to influnce where we put the line and so could not figure out the shape of the sine and instead gave a distorted wave with higher frequency.

Support vectors

The support vectors most of the time are the samples that have high influence on prediction. Normally they are the ones close to the hyperplane. Because of our transformation they are not necessarily the closest in the original space. But we can notice they are most often around the tips of the sine wave which probably has a big influence on the shape. This is because only they have a weight vector $w_i > 0$ so only they influence predictions.

Housing

In Figure 2 we see two plots. Left one shows a comparison using the polynomial kernel and right using the RBF kernel. The numbers represent the number of support vector. The upper number is the support vector count without regularization and the below is with regularization. For the second case I ran grid search to find a good lambda.

In the polynomial case the Ridge model performing better at the start but when the degree of the polynomial kernel goes over 4 the loss skyrockets. My explanation is that kernalized ridge regression is much more sensitive to outliers that are far away from most of the data. When we raise to a high power I believe we can likely create such outliers which moves the boundry far from the optimal solution. As a result we get much worse losses. On another note sometimes my matrices were not singular which also can be a problem. For me numpy resolved this issue by itself using approximations.

On the other hand because the SVR is not sensitive to outliers not close to its border becuase it ignores such samples it continues to perform similarly. The performance still becomes worse with time which is likely due to ovefitting. Interestingly the version without regularization at the end does almost as good as the one with. Still in both plots we can see the benefits of regularization as the optimized models perform much better.

In the RBF case we can see that the Ridge model is doing better compared to the SVR model. SVR is likely worse because it is trying to take into account the margin and optimize it as well. This could mean that we get higher mean squared error but the SVR should generalize better. Also depending on the data SVR can be sensitive to outliers close to its border which push it away from the optimal solution because it is trying to optimize the margin as well. This is why soft SVM and slack variables are used in practice.

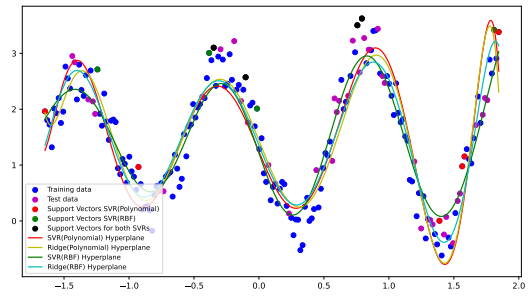

Figure 1. Sine apprxoimations

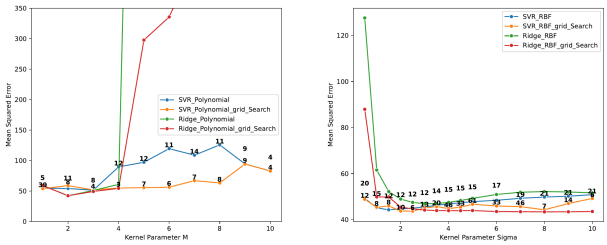

Figure 2. Comparison between models

Conclusion

In conclusion both Support Vector Regression(SVR) and Kernalized Ridge Regression have benefits and minuses. By optimizing a margin SVR should generalize better and give more accurate predictions but it can be sensitive to outliers close to the boundry which might cause it to move. In those cases Kernalized Ridge Regression can outperform it. On the otherhand since it only takes into account the support vectors SVR is less sensitive to outliers far away from the boundry. In those cases Kernalized Ridge Regression is likely to perform much worse.

References

[1] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14:199–222, 2004.