

Kernel Ridge Regression in Matlab

By Joseph Santarcangelo

This code implements Kernel Ridge Regression [1], just run main.m, there is also a function to generate some polynomial toy data and randomly partition the data into training and validation data. This document describes the class KernelRidgeRegression.

We start with the assumption that our hypothesis $h(\mathbf{x})$ is a linear combination of some basis function:

$$h(x) = \omega^T \phi(x) , \quad (1)$$

we would like to determine ω . Like regular regression we assume the difference between some target variable Γ and some deterministic function is corrupted by additive noise :

$$\Gamma = \omega^T \phi(x) + \zeta \quad (2)$$

Making the Gaussian assumption on Γ and on the parameters it can be shown using Maximum A Posteriori Estimation the we can obtain the optimal values of the vector ω by minimizing the following cost function:

$$C(\omega) = \sum_{i=1}^N \left(\Gamma_i - \omega^T \phi(x_i) \right)^2 + \frac{\lambda}{2} \omega^T \omega \quad (3)$$

It's not that difficult to show that the value of ω that minimizes (3) is given by:

$$\omega = \left(I\lambda + \sum_{i=1}^N \left(\phi(x_i) \phi(x_i)^T \right) \right)^{-1} \left(\sum_{i=1}^N \Gamma_i \phi(x_i) \right). \quad (4)$$

If the dimension of $\phi(\mathbf{x})$ is very large and for several other reasons we may want to express equation (1) in terms of Kernels. The work of [1] showed using Lagrange multipliers that equation 1 can be expressed in the form:

$$h(x) = k(x)^T (K + \lambda I)^{-1} \Gamma \quad (5)$$

Where the kernel is given by $k(\mathbf{x}, \mathbf{z}) = \phi^T(\mathbf{x}) \phi(\mathbf{z})$. The vector $\Gamma = [\Gamma_1, \Gamma_2, \dots, \Gamma_N]^T$ is the target of N training samples. The matrix \mathbf{K} is the Gram matrix with elements $(K)_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, generated from the explanatory variables from the training data. The vector $\mathbf{k}(\mathbf{x})$ whose i -th elements are in the form $(\mathbf{k}(\mathbf{x}))_i = \phi^T(\mathbf{x}) \phi(\mathbf{x}_i)$. Where \mathbf{x}_i is the i -th training sample. There are several stipulations to verify something is a valid kernel [2].

Code

To create a Kernel Regression object simply type the following:

```
KernelRegression = KernelRidgeRegression(ker, XTrain, parameters, Target, RegulationTerm);
```

	Value
Ker	Is the type of kernel ker: linear 'lin', polynomial 'poly', Radial basis function: 'rbf' and sam kenal 'sam'

X	Training data
Parameters	Linear kernels bias:b Parameters=b; $k(\mathbf{x}, \mathbf{z}) = (\mathbf{b} + \mathbf{x}^T \mathbf{z})$ Polynomial Parameters=[b d]; $k(\mathbf{x}, \mathbf{z}) = (\mathbf{b} + \mathbf{x}^T \mathbf{z})^d$ Radial basis function Parameters=[sigma]
Target	Target data $\Gamma = [\Gamma_1, \Gamma_2, \dots, \Gamma_N]^T$
RegulationTerm	λ : this value is determine imperially using cross validation

In order to produce a prediction enter the object “KernelRegression” and some validation data Xval into the method “KernelPrediction”

Prediction =KernelPrediction(KernelRegression, Xval)

The output will be given with the same length as Xval. Fig 1 shows a Polynomial function estimated with a Polynomial kernel, this can be generated from running the main function. Fig 2 shows a polynomial function estimated with a RBF kernel, it is self evident the kernel does not fit the data.

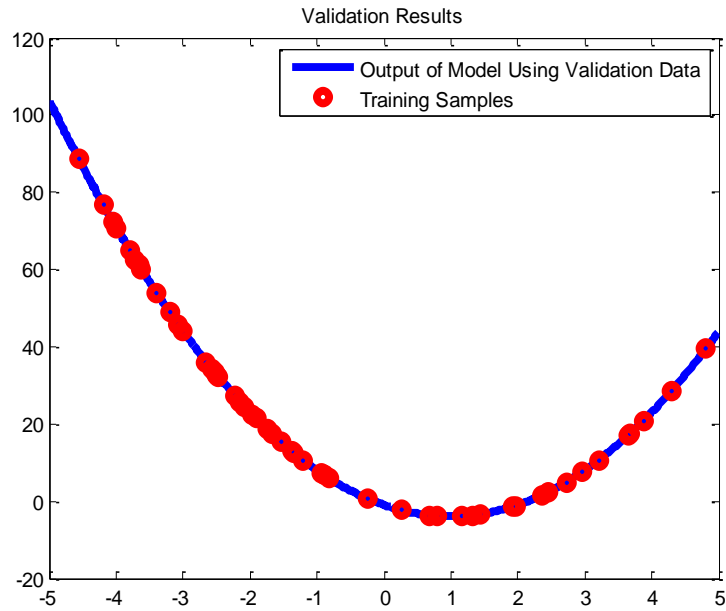


Fig 1. Polynomial function estimated with polynomial kernel.

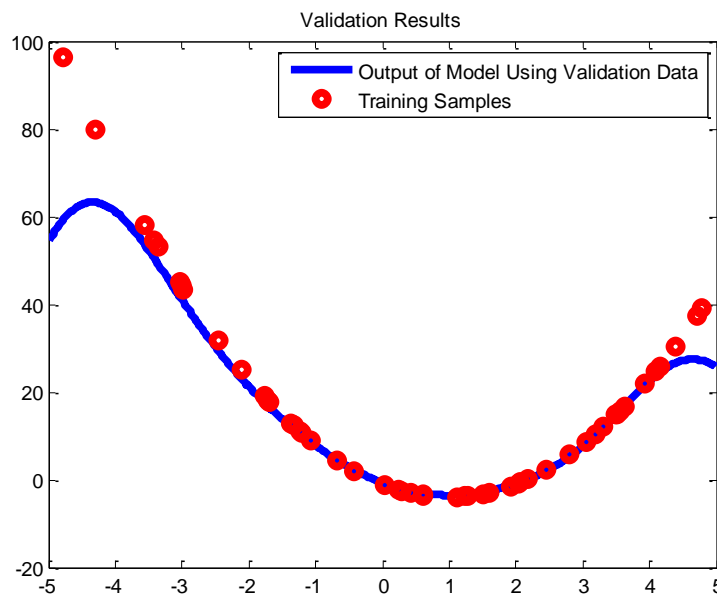


Fig 2. Polynomial function estimated with RBF kernel.

Acknowledgments

In this code I generated the kernels using a function written by Gustavo Camps-Valls and Jordi (jordi@uv.es)

References

- [1] Saunders, Craig, Alexander Gammerman, and Volodya Vovk.
"Ridge regression learning algorithm in dual variables." In (ICML-1998)
Proceedings of the 15th International Conference on Machine Learning,
pp.515-521. Morgan Kaufmann, 1998

- [2] Kung, Sun Yuan. Kernel Methods and Machine Learning.
Cambridge University Press, 2014.