Kernel ridge regression

MVE440 Statistical learning for big data

Project aim

- Simulate data as the model $\mathbf{y} = f(X, \beta) + \boldsymbol{\varepsilon}$ for $\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\varepsilon} \in \mathbb{R}^{\boldsymbol{n}}$ $\boldsymbol{and}$ $\boldsymbol{\beta} \in \mathbb{R}^{\boldsymbol{p}}$, where different f(X) were tested.
  - Data is simulated by following the instructions on how to simulate data, with the modification that f(X) takes on non-linear shapes as well as that the feature data is generated uniformly as a random sample and β is set constant to 1.
  - SNR is between 1 and 10
  - Observations n=1000
  - Features p=1
  - The data was averaged over different simulations since the data was generated randomly.

- Discuss the effect of different kernels and their hyperparameters on the methods capabilities.
  - Gaussian RBF kernel
  - Linear kernel
  - Polynomial kernel

Methods

- Setup:
  - Python
  - Library scikit-learn for machine learning
    - **from sklearn.metrics import mean_squared_error:** Calculates the mean squared error regression loss from the true and predicted values of the model.
    - **from sklearn.kernel_ridge import KernelRidge:** combines ridge regression with the kernel trick.
    - **from sklearn.model_selection import GridSearchCV:** Exhaustive search over specified parameter values for an estimator. The parameters of the estimator are optimized by cross-validated grid-search over a parameter grid.

- Methods for analysing result:
  - 5-fold cross validation was used to find optimal hyperparameter lambda in ridge regression as well as the hyperparameters for each kernel function
    - Linear kernel: c
    - Polynomial kernel: γ, r, m
    - RBF kernel: σ

  - Uniformly distributed datasets
  - Table with averaged MSE and R2 scores for the different kernels and different non-linear datasets
  - Plot that visualizes the true and estimated responses for the different kernels

The kernel functions

- The kernel functions are used to map the original non-linear dataset with non-linear into a higher dimensional space, since if the dataset isn't linearly separable in original space it however might be linearly separable in the higher dimensional space.

- **A polynomial kernel** allows us to model feature co-occurence up to the order of the polynomial (m).
  - $\boldsymbol{k(x, y) = (\gamma x^T y + r)^m}$

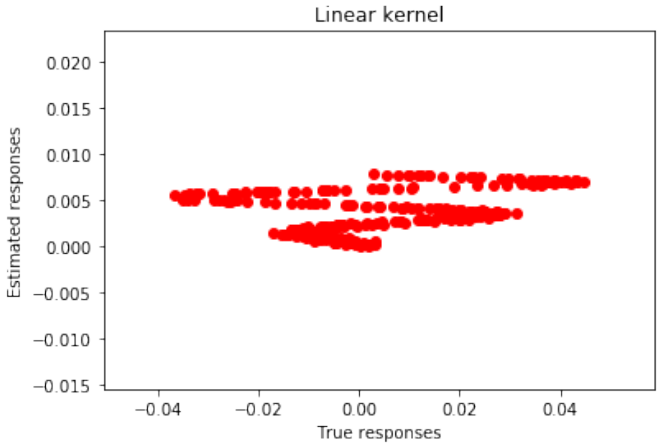
- **Gaussian radial basis kernels** allows to pick out circles (or hyperspheres)
  - Every multivariate polynomial (of any degree) can be used as a linear predictor with this kernel.
  - $\boldsymbol{k(x, y) = \frac{\exp(-||x-y||_2^2)}{2\sigma^2}}$

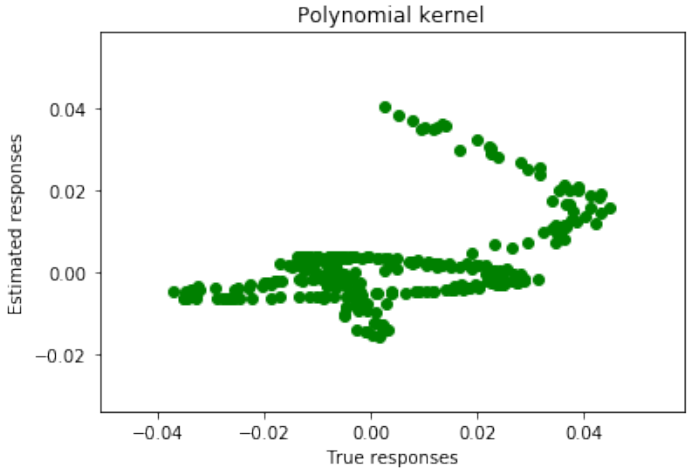
- **Linear kernels** allows only to pick out lines.
  - The Linear kernel is the simplest kernel function.
  - Linear separators in the high-dimensional space correspond to highly nonlinear decision boundaries in input space.
  - $\boldsymbol{k(x, y) = x^T y + c}$

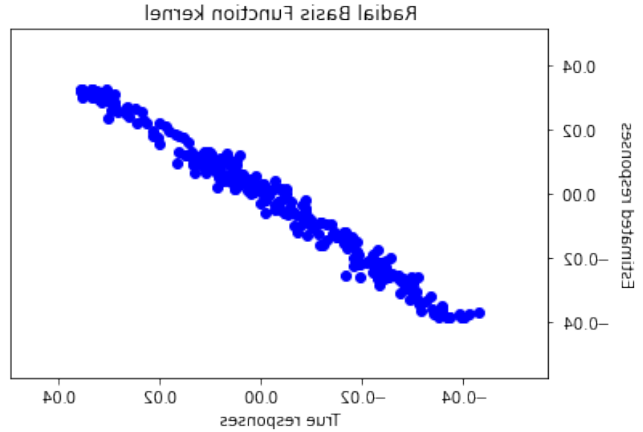
Data simulated as:

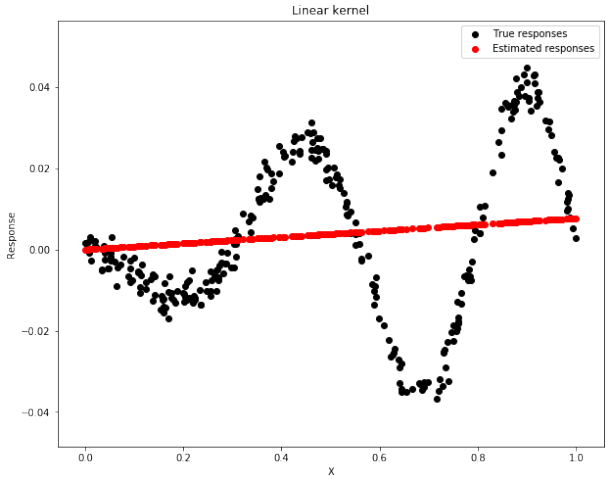
$$f(X) = (sinc(X^2 + 3X) * sin(X)^2)$$

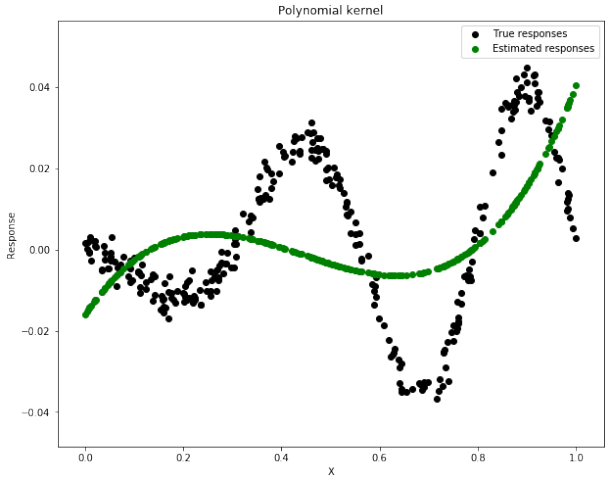
- SNR=8.1
- Plots simulated after one run

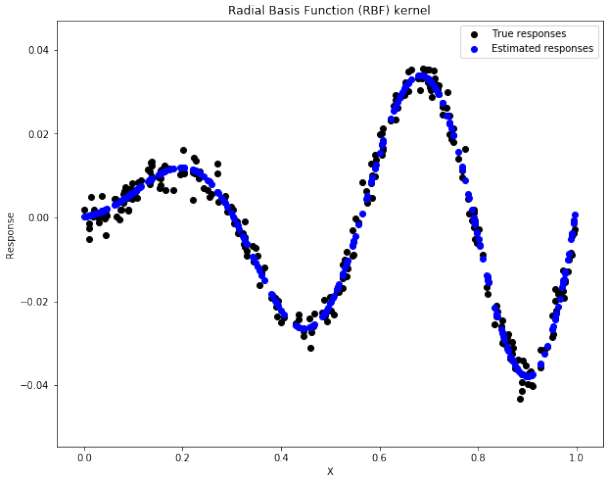






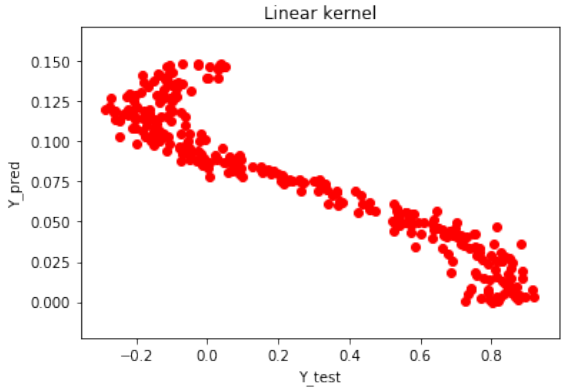


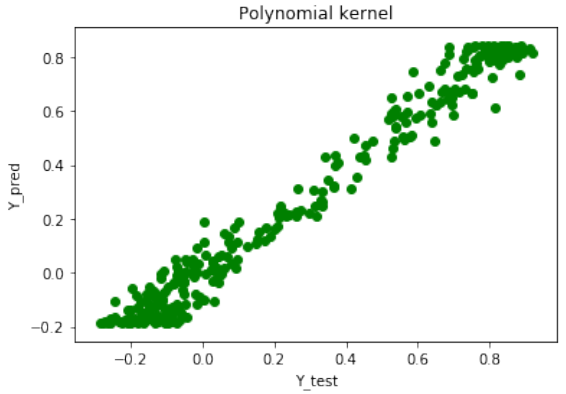


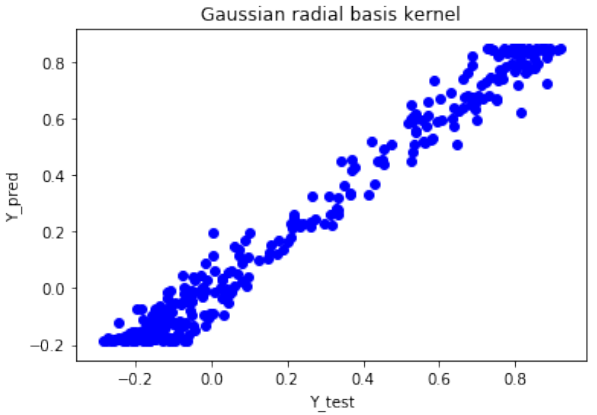


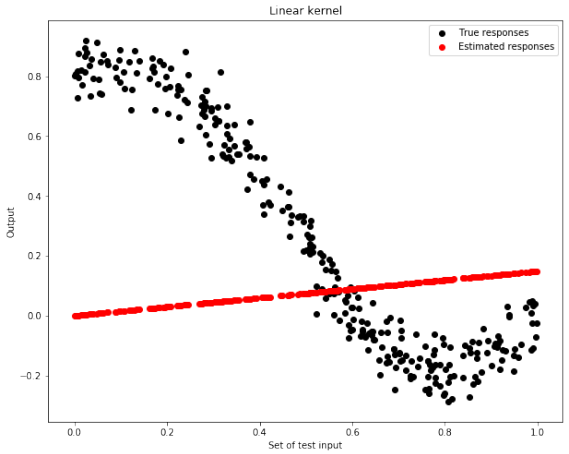
Data simulated as $\mathrm{f(X)} = sinc(X^3 + X)$

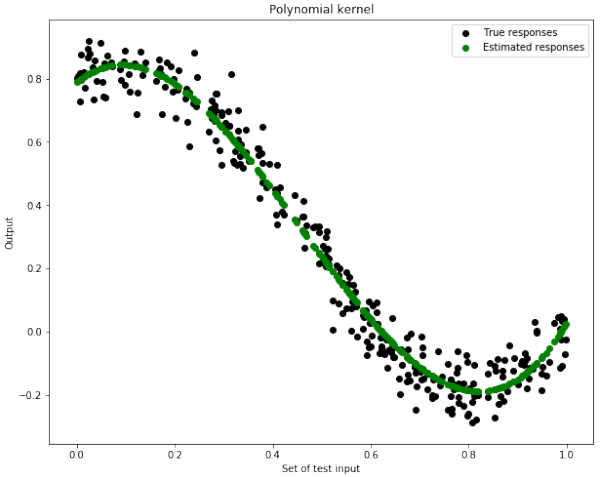
- SNR=8.2
- Plots simulated after one run

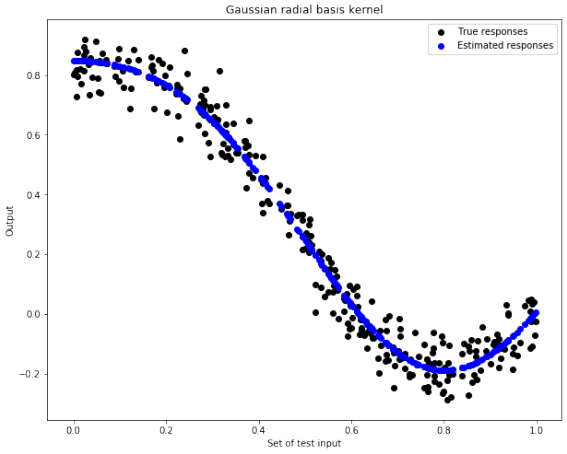






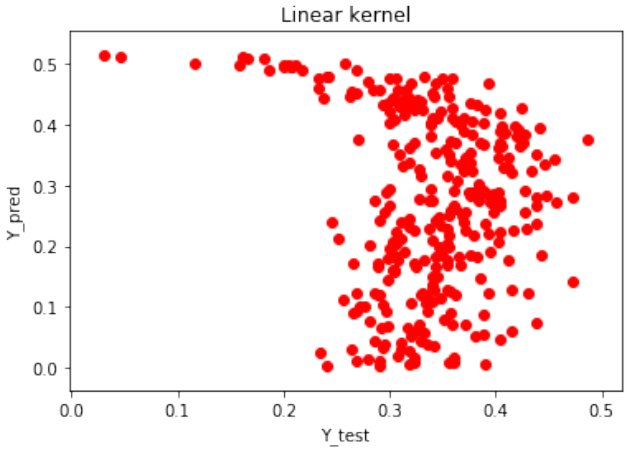


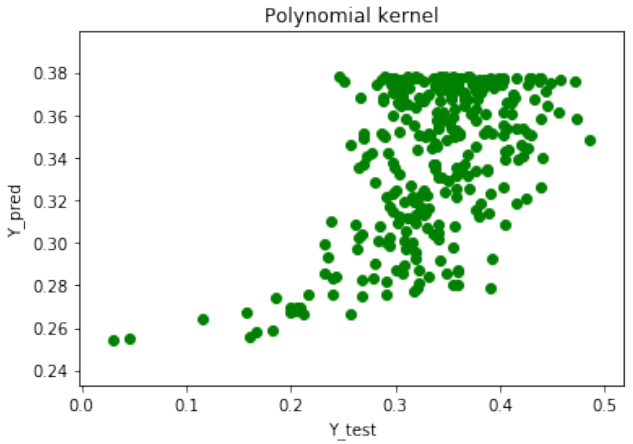


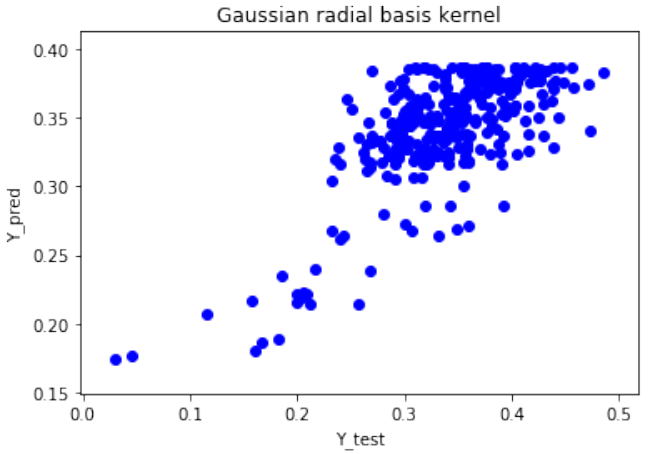


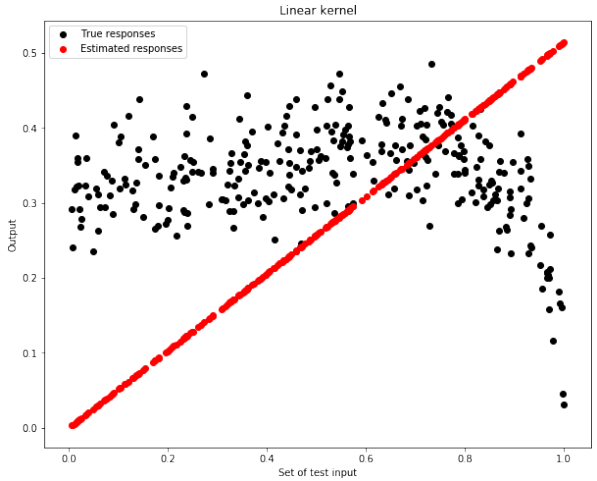
Data simulated as $f(X) = sin(e^{X^2})$

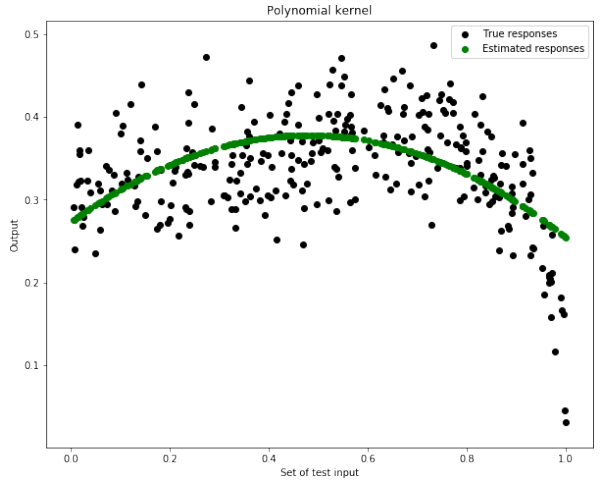
- SNR=7.9
- Plots simulated after one run

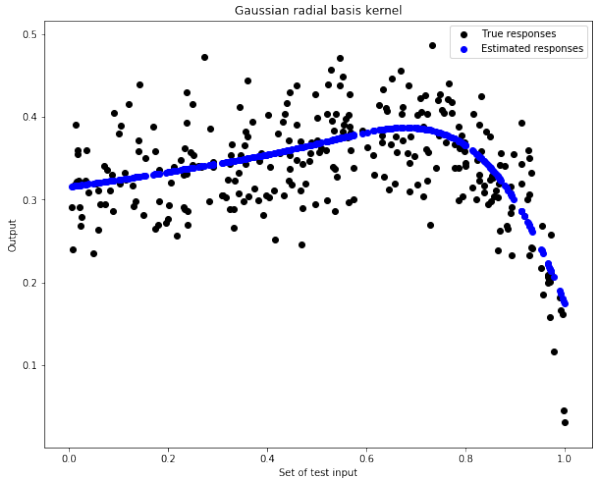












Averaged MSE and R2 score for different kernels and datasets

- The MSE was averaged over different simulations for each dataset and kernel

Kernel	$f(X) = (sinc(X^2 + 3X) * sin(X)^2)$, **MSE**	**R2 score**	$f(X) = sinc(X^3 + X)$ **MSE**	**R2 score**	$f(X) = sin(e^{X^2})$ **MSE**	**R2 score**
Linear	$6.75 * 10^{-4}$	0.027	0.095	-0.496	0.111	-8.838
Polynomial	$5.33 * 10^{-4}$	0.074	0.002	0.974	0.008	0.26
Gaussian	$1.13 * 10^{-5}$	0.973	0.002	0.976	0.006	0.463

*If the fit is actually worse than just fitting a horizontal line then R2 is negative

Discussion

- The gaussian RBF kernel and the polynomial kernel generally perform equally good. This can bee seen by simply investigating the equations of these kernels, which explains why they perform well on non-linear datasets in contrast to the linear kernel.

- The complexity of the RBF gaussian kernel increases with the size of data, whereas the polynomial will be saturated after some point, thereby the polynomial kernel tend to perform equally well as the RBF gaussian kernel when the dataset isn't to complicated, i.e. when its still possible to accordingly map the data with the polynomial kernel.
  - Can be seen at the averaged MSE and R2 score
  - MSE is generally lowest for the Gaussian, then the polynomial and lowest for the linear kernel and vice versa for the R2 score.

- For the data set with $\mathrm{f(X)} = sinc(X^3 + X)$, the polynomial kernel seems to perform equally well as the gaussian RBF kernel. However, with $\mathrm{f(X)} = sin\left(e^{X^2}\right)$ and $\mathrm{f(X)} = (sinc(X^2 + 3X) * sin(X)^2)$, the gaussian RBF seems to outperform the polynomial kernel as well. The plots visualizes that the gaussian RBF kernel captures the shape of the data better then the polynomial and the linear for these setups.

Conclusion

- Linear and polynomial kernels proved to be less time consuming when running the code, but however provided less accuracy than the radial basis kernels.
- The gaussian RBF kernel performs best and seems to be more flexible than the linear or polynomial kernels in that you can model a whole lot more functions with its function space since its squared exponential.
  - The gaussian RBF kernel might thereby be a good choice when a complex model is assumed or when there is not much known about the data.
- However, if you know (or assumes) that the true function can be well approximated by a linear or polynomial function, then a linear or polynomial kernel seems to be better. The linear and polynomial will probably be fitted using a lot less data then the squared exponential, i.e. less time consuming due to being less computational heavy.