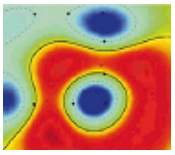
Technische Universität Berlin



Problem Set 3: Kernel Ridge Regression, Cross-Validation

Report Machine Learning Lab Course
Fachgebiet Maschinelles Lernen
Prof. Dr. Klaus-Robert Müller
Fakultät IV Elektrotechnik und Informatik
Technische Universität Berlin

submitted by

Budi Yanto

Instructor: Daniel Bartz
Felix Brockherde

Matrikelnummer: 308819
Email: budiyanto@mailbox.tu-berlin.de

Chapter 1

Implementation

In this problem set, *Kernel Ridge Regression (KRR)* and *Cross-Validation (CV)* should be implemented. Section 1.1 and Section 1.2 describe the implementation of *KRR* and *CV*, respectively.

1.1 Assignment 1: Cross-Validation

In this assignment, *Cross-Validation* should be implemented as follows:

```
method = cv(X, y, method, {param_name:  value_range, ...},
           loss_function, nfolds, nrepetitions)
```

The function *cv* returns *method*, which has been trained with the optimal parameter values. In addition, it also has an attribute *method.cvloss* containing the cross validated loss.

Figure 1.1 shows the test result of CV. The CV with fixed regularization took about 5.6 seconds to finish and obtained $kernelparameter = 0.5994$ and $regularizationparameter = 0.0774$ as the optimal parameter values. On the other hand, the CV with efficient LOOCV took about 3.3 seconds to finish and obtained $kernelparameter = 0.5994$ and $regularizationparameter = 1.8$ as the optimal parameter values

1.2 Assignment 2: Kernel Ridge Regression

In the second implementation assignment, *Kernel-Ridge-Regression* should be implemented as a class:

```
class krr(X, y, kernel, kernelparameter, regularization)
```

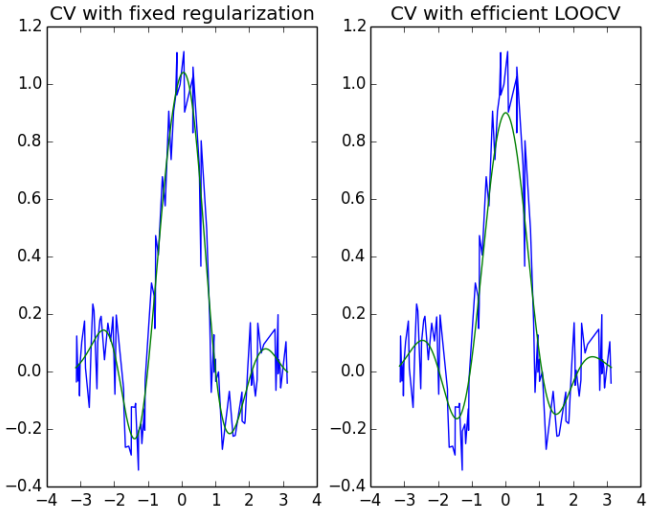


Figure 1.1: CV test

The *krr* class has the functions `fit(X, y, kernel, kernelparameter, regularization)` and `predict(X)`. The *linear*, *polynomial* and *gaussian* kernel should also be implemented.

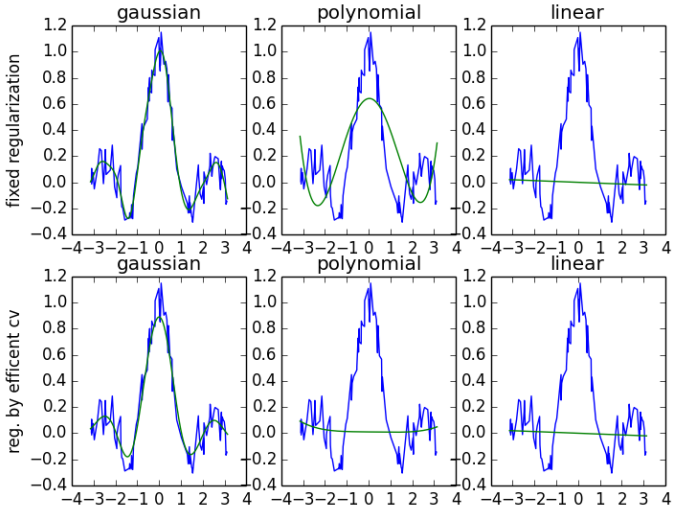


Figure 1.2: KRR test

Figure 1.2 shows the test result of *krr*. The *krr* was executed using fixed regularization parameter and regularization parameter obtained by efficient LOOCV. By *krr* with

fixed regularization parameter, the test results look promising, but by *krr* with efficient LOOCV, the result of polynomial kernel did not look very good. The reason for that is i got a very high mean of the eigenvalues of the kernel, so that the regularization parameter is also very high. Maybe my polynomial kernel implementation is wrong, but i cannot find out where the mistake is. For the gaussian kernel, the regularization parameter of 1.8 was obtained, while for the polynomial and linear kernel, the regularization parameters was 1349.74 and 3.961, respectively.

Chapter 2

Application

2.1 Assignment 3: ROC Curve

In this assignment, the ROC curve of a simple 1D-toy data set with two classes should be plotted. Each class is normal with standard deviation one and identical priors. Figure 2.1 depicts the ROC curve of the data set. Four sample sizes are used to generate the ROC curve: 10, 100, 1000 and 10000.

2.2 Assignment 4: Kernel Ridge Regression

In this assignment, the *krr* should be applied to five data sets: *banana*, *diabetis*, *flare-solar*, *image* and *ringnorm*. *krr* should be applied to each data set using efficient cross-validation and then finds the best classifier by cross-validation. The normal CV takes a longer time than the efficient LOOCV to optimize the regularization, because it has to cross-validate all of the folds and calculate the inverse matrix. On the other hand, the efficient LOOCV only do one inverse in order to get the optimal regularization parameter. Furthermore, the inverse of efficient LOOCV is only an inverse of a diagonal matrix which can be easily calculated by inverting the diagonal. Table 2.1 shows the optimal parameters for each data set.

From the experiment, it turned out that gaussian kernel is the optimal kernel for all data sets. But i think the regularization parameter obtained by the efficient LOOCV was wrong, because it is always 1.8 for all data sets. I think the reason for that is because the eigenvalues of the kernel matrix K used for c in efficient LOOCV are very small values, so that it always return $mean = 1$. Unfortunately, i cannot find where the mistake is. The whole experiment took about 6 hours. The longest CV execution is by *image* data set which took about 3.5 hours.

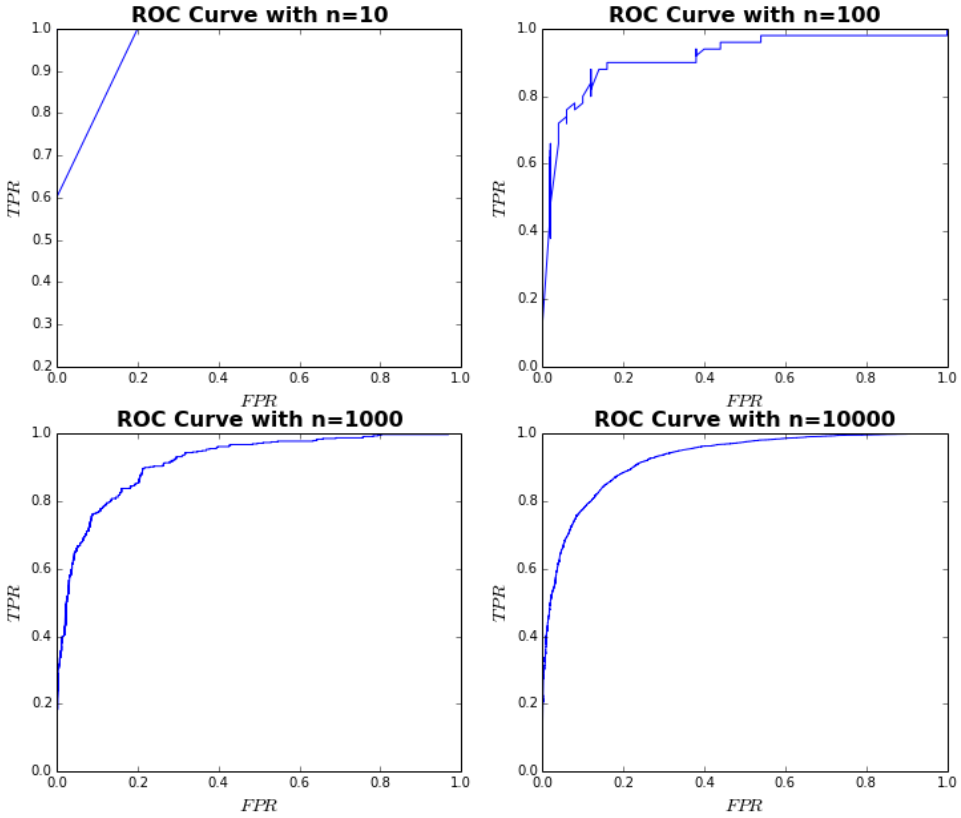


Figure 2.1: ROC curve with different sample sizes

Dataset	Kernel	Kernelparameter	Cvloss	Regularization	Time (in seconds)
Banana	Gaussian	0.215	0.099	1.80	272.435
Diabetis	Gaussian	4.641	0.243	1.80	584.214
Flare-Solar	Gaussian	4.641	0.354	1.80	949.138
Image	Gaussian	0.599	0.029	1.80	12544.090
Ringnorm	Gaussian	0.077	0.046	1.80	291.681

Table 2.1: The optimal parameters for each data set