Machine Learning	Prof. Dr. Bernt Schiele
	Dr. Paul Swoboda & Dr. Gerard Pons-Moll
Solution 3 - 05.11.2018	due: 12.11.2018, 14:15

Exercise 7 - Regression in Practice

We consider the regression problem where the input $\mathrm{X} \in \mathbb{R}^d$ and the output $\mathrm{Y} \in \mathbb{R}$. We use either L_1 or L_2-loss.

a. **(2 points)** In the following you have to implement least squares and ridge regression (both L_2-loss).

- `w = LeastSquares(Designmatrix,Y):`
  - input: design matrix $\Phi \in \mathbb{R}^{n \times D}$ and the outputs $Y \in \mathbb{R}^n$ (column vector)
  - output: weight vector w of least squares regression as column vector
- `w = RidgeRegression(Designmatrix,Y,Lambda):`
  - input: the design matrix $\Phi \in \mathbb{R}^{n \times D}$, the outputs $Y \in \mathbb{R}^n$ (column vector), and the regularization parameter $\lambda \in \mathbb{R}^+$
  - output: weight vector w of ridge regression as column vector. Use the non-normalized version: $w = (\Phi^T \Phi + \lambda \mathbb{1}_D)^{-1} \Phi^T Y$.

Note that the matlab code for L_1-loss (with and without L_2-regularizer) is provided in the zip-file. It requires the installation of CVX - you can find a link on the course webpage.

b. **(1 Point)** Write a Matlab function `Basis`(X, k):

- input: the input data matrix $X \in \mathbb{R}^{n \times 1}$ and the maximal frequency k of the Fourier basis.
- output: the design matrix $\Phi \in \mathbb{R}^{n \times (2k+1)}$ using the Fourier basis functions:

$$\phi_0(x) = 1, \qquad \phi_{2l-1}(x) = \frac{1}{l}\cos(2\pi l x), \qquad \phi_{2l}(x) = \frac{1}{l}\sin(2\pi l x), \quad l = 1, \ldots, k.$$

c. In the first example we have only one feature, thus we want to learn a function $f : \mathbb{R} \to \mathbb{R}$. First plot the training data (`plot(Xtrain, Ytrain,'.');`).

- **(2 Points)** Which loss function (L_1 or L_2) is more appropriate for this kind of data ? Justify this by checking the data plot. Use in the next part only the regression method with your chosen loss (that is either ridge regression or L_1-loss with L_2-regularizer).
- **(4 Points)** Use the basis functions with $k = 1, 2, 3, 5, 10, 15, 20$ from part b) to fit the regularized version of the loss chosen in the previous part. Use regularization parameter $\lambda = 10$. Plot the resulting functions (use `x = 0 : 0.01 : 1`) for all values of k together with the training data,

$$f_k(x) = \left\langle \phi(x), w^k \right\rangle = \sum_{i=1}^{2k+1} w_i^k \phi_i(x).$$

Save the plots using the command `saveas(gcf, 'PlotFunctions', 'png')`.

Compute the loss, that is

$$\frac{1}{n} \sum_{i=1}^{n} L(Y_i, f(X_i)),$$

on the training and test data (variable names `trainloss` and `testloss`) and plot training and test loss as a function of k. Save the plots using the command `saveas(gcf, 'PlotLoss', 'png')`. Save your training and test loss in a file `LossFirstEx`, use

`save LossFirstEx trainloss testloss`

Repeat the same for $\lambda = 0$ (unregularized version) - append a 0 to all filenames e.g. `LossFirstEx0`.

How does increasing k affect the estimated functions f_k ? What is the difference in terms of k of the regularized and unregularized regression method. The last two questions have to be answered on paper.

d. The second example is a real dataset. The task is to predict the total number of violent crimes per 100K population (output variable $Y \in \mathbb{R}$) from a set of features (input variables $X \in \mathbb{R}^{99}$) capturing all sorts of properties of the cities and their population.

- **(2 Points**) Use a linear design with an offset, that is $f(x) = \langle w, x \rangle + b$, (add a feature which is 1 for every data point, $\texttt{X} = [\texttt{X}, \mathbf{1}(\texttt{size}(\texttt{X}, 1), 1)]$) and fit the data using least squares regression. Compute the training loss, that is $\frac{1}{n}\sum_{i=1}^{n} \|Y - X * w\|_2^2$, on the training set. Save this `trainloss` with `saveLossSecondExtrainloss`.
- **(2 Points)** You are now free to use any set of basis functions and any regression method. Write a function `Prediction2(X)` which given a set of testpoints - a matrix $X \in \mathbb{R}^{n \times 99}$ outputs the predictions of your chosen learning method as a column vector $f \in \mathbb{R}^{n \times 1}$. For the best results on our hidden test set (which have to be better than the results of linear least squares) we have the following prizes
  1. **(10 Bonus Points)** for the winner
  2. **(5 Bonus Points)** for the second best prediction
  3. **(3 Bonus Points)** for the third best prediction

Submission instructions

- We accept both handwritten and electronic submissions. So you can choose what is more convenient for you. In any case, you should specify full names and immatriculation IDs of all team members. Obviously, programming tasks you can submit only electronically.
- Handwritten submissions should be submitted in the lecture hall of Monday's lecture (before the lecture starts).
- Electronic submissions should be zipped, containing the m-files (`Basis` etc.), your plots (png files) and the matlab data files (.mat) and emailed to the corresponding tutor:
  a. Apratim Bhattacharyya (Wednesday 8-10): abhattac@mpi-inf.mpg.de
  b. Maksym Andriushchenko (Thursday 8-10): s8mmandr@stud.uni-saarland.de
  c. Max Losch (Friday 16-18): mlosch@mpi-inf.mpg.de

  If not all 3 students belong to the same tutorial group, then you should email your submission to **only** one tutor (e.g. to the tutor of the first author of your homework), so please do not put other tutors in copy of the email.

  The email subject must have the following form: "[ML18/19 Exercise] Sheet X", where X is the number of the current exercise sheet. Then please specify in the email full names and immatriculation IDs of all team members. Then please attach all your files as a single zip archive, which consists of your immatriculation IDs, e.g. "2561234_2561235_2561236.zip".
- Reminder: you should submit in groups of 3. Otherwise, we will later on merge the groups smaller than 3 students.

Hints:

- In order to have several plots in one figure you have to use
  `figure`, `hold on`
  ... all your plotting commands ...
  `hold off`
- `gcf` is a handle to the current figure (save the figure just after it was created).
- In order to distinguish the curves for different values of k draw them in different colors using:
  `Colors` = `jet`(`NVals`),
  `for k` = 1 : `NVals`, `plot`(`x`, `Output`(:, `k`), 'linestyle', ' − ', 'color', `Colors`(`k`, :)), `end`
  where `Output`(:, `k`) is a matrix containing the estimated function values at `x`.
- `norm`(`x`) computes the Euclidean norm of a vector `x`.
- Linear system, `Ax` = `b`, can be solved in Matlab using the backslash operator `x` = `A\b`.
- More details on the features for the second task can be found in the data-file.

Solution:

a.
- Least Squares:
```
function w=LinearLeastSquares(X,y)
% computes the coefficients w of the linear least squares matrix
% given:
% X: design matrix
% y:  outputs

w = (X'*X)\(X'*y);
```

- Ridge Regression:
```
function w=RidgeRegression(X,y,lambda)
% computes the coefficients w for ridge regression
% given:
% X: design matrix
% y:  outputs
% lambda:  regularization parameter

num = size(X,1);
dim = size(X,2);

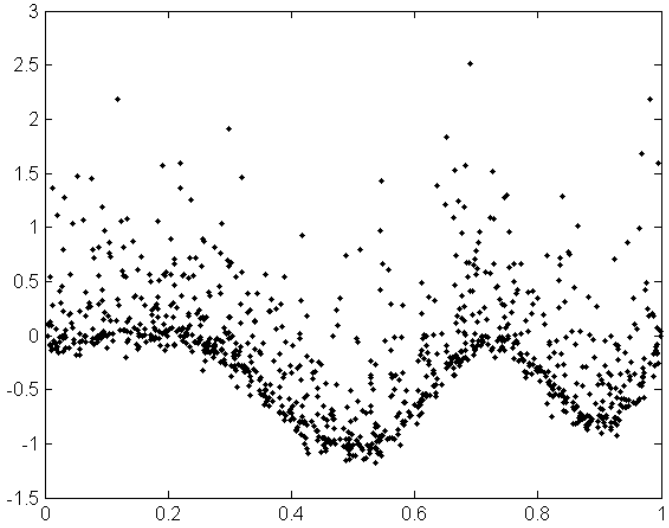
w = (X'*X + lambda *eye(dim))\(X'*y);
```

b. The Fourier-basis (normalized)

- ```
  function Phi=Basis(X,k)
  % computes the design matrix for a Fourier basis
  % X: input matrix
  % k:  number of Fourier basis functions (actually 2k+1 functions)
  %
  % output:  design matrix X
  ```
  ```

```
Phi(:,1) = ones(size(X,1),1);
for i=1:k
Phi(:,2*i) = 1/i*cos(2*pi*i*X);
Phi(:,2*i+1) = 1/i*sin(2*pi*i*X);
end
```

c. • We first plot the data:



We observe that the conditional distribution $p(y \mid x)$ is very skewed but still it appears to be uni-modal (one maximum). As we have heavy tails on the "upper side", estimation with least squares loss leads will not yield a good result as the mean of $p(y|x)$ is not a good "representative", whereas the median of $p(y|x)$ is a much better choice and thus the L_1-loss is the method of choice here.
The data has actually been generated like this

$$y = f(x) + \varepsilon,$$

where $f(x) = \sin(2\pi x^2)\cos(2\pi x)$ and the noise ε has the density

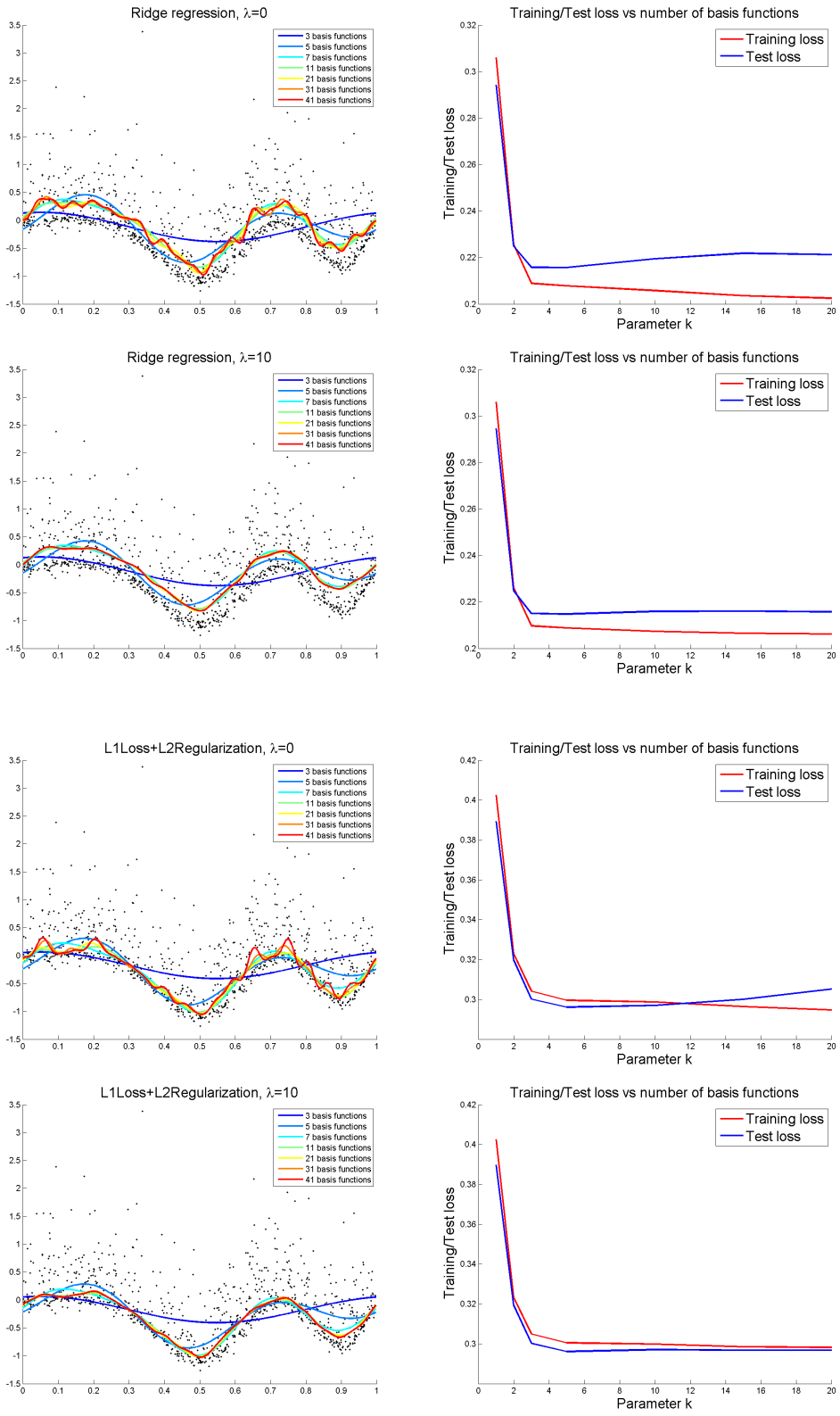
$$p(\varepsilon) = \begin{cases} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\varepsilon^2}{2\sigma^2}}, & \text{for} \quad \varepsilon \leq 0, \text{where} \quad \sigma = 0.1, \\ \frac{1}{2}\lambda e^{-\lambda\varepsilon}, & \text{for} \quad \varepsilon > 0, \text{where} \quad \lambda = 2. \end{cases}.$$

This is a mixture of a zero-mean Gaussian truncated at zero and an exponential distribution both with weight $\frac{1}{2}$. Note that the median of $p(\varepsilon)$ is at zero ! (Half a Gaussian + Half Exponential distribution). However, the mean (expectation) is

$$\mathbb{E}[\varepsilon] = -\frac{\sigma}{\sqrt{2\pi}} + \frac{1}{2\lambda} \approx 0.21$$

Here we use that the mean of zero-mean Gaussian random variable truncated at zero is given by $-\frac{2\sigma}{\sqrt{2\pi}}$ and the mean of a random variable with exponential distribution with parameter λ is $\frac{1}{\lambda}$

- We provide all plots together so that one can see the difference of L_2-loss and L_1-loss based methods.

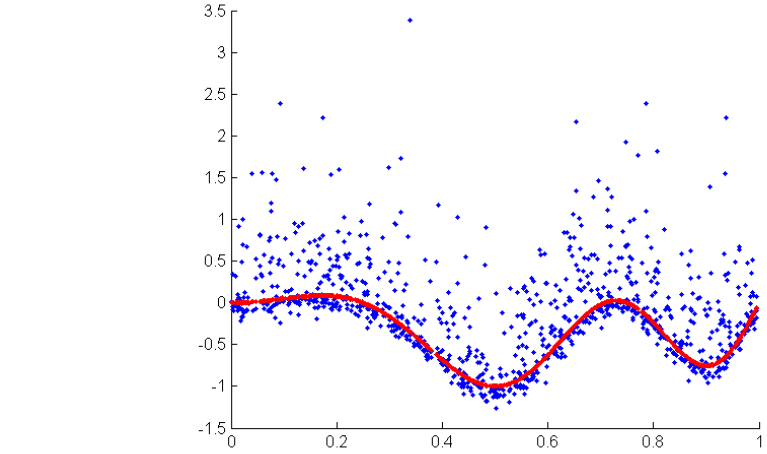


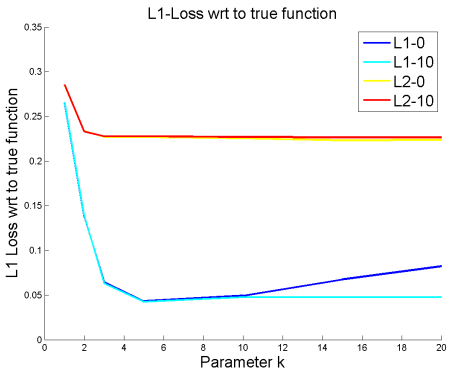
For $\lambda = 0$ the training loss is monotonically decreasing as a function of k, as we can just fit the data better once we have more basis functions. The problem is that at some

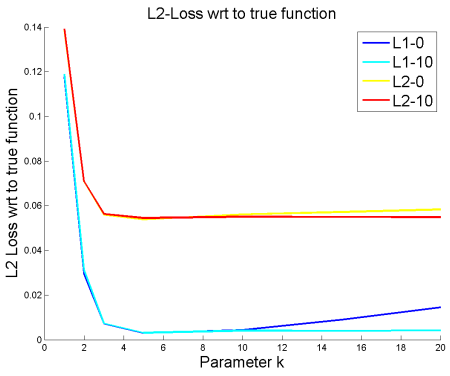
point the additional flexibility in our model is just used to fit the noise. The training loss of the regularized version is also monotonically decreasing (which is also typical) but this is not guaranteed due to the regularizer.
The test loss is increasing again with k for the pure fitting methods without regularization (as the additional flexibility is used to fit the noise - for large k the estimated function contains high frequence components). The regularized versions only slightly overfit with increasing k, basically the test loss is constant with increasing k. The reason is that while the training loss is decreasing using high frequency components, the high frequency components are heavily penalized by the regularizer.

Note, that for the L_2-loss based methods, the fitted function is much higher than the one fitted with L_1-loss. The reason is that with L_2-loss we are trying to estimate the mean of $p(y|x)$ whereas with L_1-loss we are trying to estimate the median of $p(y|x)$. As $p(y|x)$ is heavily skewed, median and mean do not agree. Moreover, the noise distribution has been generated so that the "true function" is equal to $p(y|x)$.

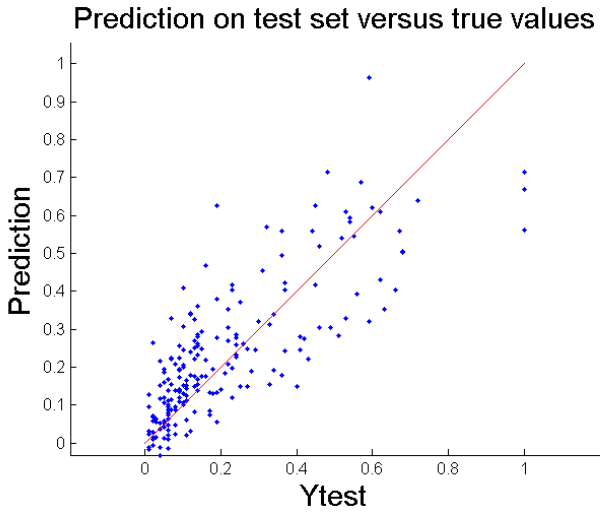
Due to the above mentioned fact, the fitted functions for L_1-loss are much better estimators of the true function than the L_2-loss as can be seen in the following plots, where we plot the L_1-loss resp. L_2-loss of our estimated functions with respect to the true function.







d. The training loss achieved by least squares is given as 0.0168. The corresponding test loss on the hidden test set is 0.0159. This already not too bad. In the plot below we show the predicted crime rate versus the true crime rate. We see that for small true crime rate values we tend to overestimate the crime rate, whereas for large true crime rate values we tend to underestimate them.



e. I optimized on the test set which is clearly not allowed. We learn later methods how to do proper model selection e.g. via cross-validation. My best method uses just L_1-loss with the design in matlab notation $\texttt{Phi} = [\texttt{X}, \texttt{ones}(\texttt{size}(\texttt{X}, 1), 1), \texttt{X}.^2]$. It achieves a L_2-test loss of 0.0137 which is quite a bit better, the corresponding plot of predicted versus true values is shown below.