Online Statistical Computation in the Feed database

Facharbeit im Nebenfach Informatik

vorgelegt

von

Zafer Adıgüzel
St. Gallen, Schweiz
Matrikelnummer

Angefertigt am
Institut für Informatik
der Universität Zürich
Prof. Dr. M. Böhlen

Betreuer: Andrej Taliun
Abgabe der Arbeit: 15.03.2011

Table of Contents

Introduction

In this project we deal with missing data values to be calculated in a database by an appropriate choice of a statistical regression and the evaluation of a database query performance. The database contains values of measurements of animal feed that have been made by specific data. The number of measurements is the main factor in the evaluation of food quality. But since chemical analysis is long and costly, we usually tend to make a sufficient number of measurements only for a small number of components. To perform a good analysis of the mixture of feed, it is essential to look at all individual components, even when they are not measured. The regression method, currently just calculated outside the database and for a limited time period, helps us to fill these missing data in the food database. We have to consider that correlated components can change with time: correlated components become uncorrelated and correlation parameters change. Therefore we consider both a linear and a non-linear regression method, kernel regression, to find correlated components, and to compare the two methods with each other. Assuming correlated components are known and parameters of linear regression are stored in the database, we introduce an ER model as a design for our feed database, translate it into a SQL schema and evaluate our SQL queries.

Database “Analyses of Barley 1992-2009”

Before we can calculate any statistical regression in the database, we have to take a closer look at it. If we consider the food database “Analyses of Barley 1992-2009” in more detail, it is obvious that there are no intrinsic values of the nutrients 'OS', 'RPT', 'RPL', 'RLPs', 'RLGC', 'RFB' and 'BE'. Therefore we can not determine the values of these nutrients, since we have no fundamental data.

The nutrient 'TSO' has only one single entry with the component “Gerste undef” from18th May 2004. The nutrients 'CA', 'SE' and 'NA' have very inaccurate measurements in the database:

- For example, the nutrient 'CA', in the components
  - “Orge entier”, “orge moulu”, “Orge APLATIE” (17th line),
  - “Orge APLATIE 1” (18th line),
  - “Orge APLATIE 1” (26th line), and
  - “Gerste” (27th line)

  has the entry "<1.0".

- With the nutrient 'SE' the measurement of the component “Gerste undef” from 2002, has three entries “<20”, which is a rather useless piece of information.

- The nutrient 'NA' contains similar information: in three entries of the component “Gerste undef” from 23rd of March 2005 we have “<.20” and the same in the last three entries in the component “Gerste undef” from 17th of August 2005.

Therefore, we cannot fill in the values of these nutrients, since we have no baseline data.

The lines 86-91 contain useful overall information for each column:

- Line 86 contains the number of component entries for a specific nutrient.
- Line 87 contains the mean value of all component entries for a specific nutrient.
- Line 88 calculates an estimate of the standard deviation based on all the entries.
- Lines 89 and 90 show the minimal and maximal component value respectively.
- Line 91 contains the median value for a specific nutrient.

Linear Regression and Kernel-Regression

By means of regression we can examine a possible correlation in our food database, concerning the relationships between the measurements. We assume that there is a connection between the nutrients. In the following we have the linear regression and the non-linear kernel regression. The common factor in both approaches is the assumption that the function is not continuously present, but only a measured pair of values $(x_i, y_i)_{i=1,\dots,n}$ are at hand, whereas y_i describes the measurement at the parameter point x_i.

Linear Regression

In the linear regression, the goal is to find an affine ("linear") figure

$$f(x) = m*x+b$$

which approaches the pair of values as near as possible. Mathematically, this means that the root mean square becomes as small as possible. For a precise representation see [Sauter 10]. The implementation of the regression in *Matlab* is dealt with in the QR-method taken from [Sauter 10]. What is important for us is the assignment of the both coefficients '*m*' and '*b*' in the affine function. For this we prepare a text file in format *'dat'*, that contains the pairs of value in two columns and as an example it could appear as follows:

901.92	25.98
874.82	24.72
863.20	27.56
927.15	26.16
919.10	25.02
876.70	30.57
896.70	26.54
......	......

The first column consists of the parameter points, the *x*-values, and the second column of the measuring points, the *y*-values. After the file is loaded we save the contents into variables. To calculate the coefficients, we solve it by means of the QR-method, which we implemented in the functions *qrsolve_er* and *qrsolve*[1] and save the coefficients in a variable named *coeff*. The corresponding M-file *linreg.m* consists of the following *Matlab* commands:

```
K = load('tsl_ra.dat');
H = K(:,1);          %% 1. Spalte: Parameterpunkte
g = K(:,2);          %% 2. Spalte: Wertepunkte
coeff = qrsolve_er(H,g);
x = min(H):0.1:max(H);
y = coeff(1)+coeff(2)*x;

plot(H,g,'bx',x,y,'r-'); grid
```

That means we can retrieve the coefficients *coeff (1)* and *coeff (2)* and thereby calculate the missing *y*-values for the appropriate components by simple insertion. Now we only have to decide where we want to assume a linear relationship. After that we simply change the file name '*tsl_ra.dat*' in the M-file *linreg.m* to the file name where our pairs of values are and the calculation procedure is the same as before.

1 See Appendix B

Kernel-Regression

In the study of linear regression we assume our data to be linear. But if we know or suppose that the data is *not* linear, we need a corresponding model, which approximates our data locally in a non-linear way. For this we use Kernel-Regression. The advantage is that it requires no distribution of the data. A set of identical weighted function called Kernel local to each parameter point is assigned, based on distance from the parameter point. So this means by putting the kernel at our parameter point x_i, we can extend the parameter values at a certain small step. There are some different ways to calculate the kernel. The formula of *Gaussian Kernel* is

$$K_\alpha(x, x_i) = \exp(-\frac{(x-x_i)^2}{2 \cdot \alpha^2})$$

where x_i is our parameter point, x the point near to x_i and α is the kernel width.[2] Then the estimated value y_j at x_j is given by the following Kernel Regression formula, known as the *Nadaraya-Watson kernel weighted average*:

$$y_j = \frac{\sum_{i=1}^{n} w_i \cdot K_\alpha(x_j, x_i)}{\sum_{i=1}^{n} K_\alpha(x_j, x_i)}$$

Or in words: "The nominator of the Kernel Regression formula is array sum product of kernel and weight, while the denominator is just sum of kernel value at domain x_j for all data point x_i."[3] The weights for each kernel have to minimize the sum of square error. See [Teknomo 07] for how this is done. There are also other kernel estimators, such as *Priestley-Chao* or *Gasser-Müller.*

To calculate the regression in *Matlab* we use a "Kernel Regression Toolbox". We save the paired values from our data file into the vectors x and y as before and start the calculation by the *Matlab* command `kern(x,y)`. Then we start to find optimized parameters for the regression, for example optimal bandwidth. After finding them, we have the results and we are then able to visualize them. To extract any numbers from the results, we plot the figure and open up the property editor in the menu "View". After clicking on "More properties..." a new window pops up and in the rows "XData" and "YData" we can find the actual values on the figure. An alternative is to use the "Data Cursor" from the "Tools" menu. See [Koláček 09] for further details.

Results

The calculated values are plotted in *Matlab*. The following is a table of five assumed linear relationships:

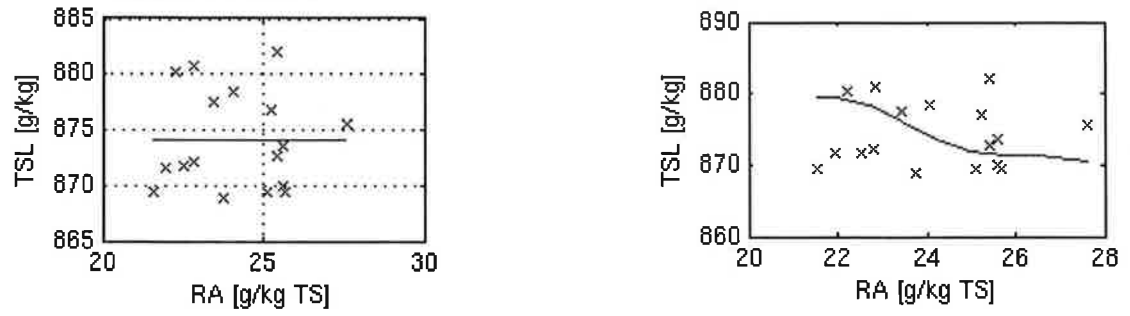
Missing values for nutrient	*Basis for regression is nutrient*	*Analysis made for the year(s)*
TSL	RA	2005, 2008
TSO	TSL	2005
RP	RA	2005, 2008
NA	P	2008
Co	SE	2008

2 From [Teknomo 07] about kernel width: "Wider kernel bandwidth will span to larger domain. You can imagine kernel width as the width of a window center at the data point and give weighting value to any points located in the window. These weights will be used as local average for all points within that window."

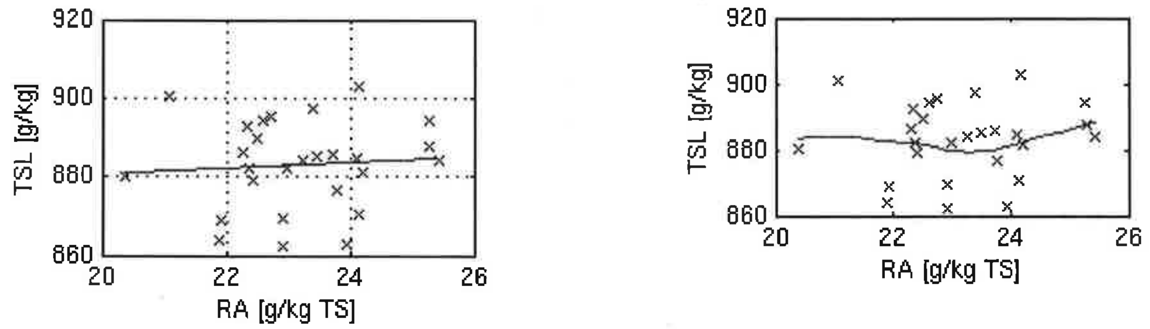
3 Taken from [Teknomo 07]

Below are the graphs, comparing the linear regression on the left and the kernel regression on the right side.

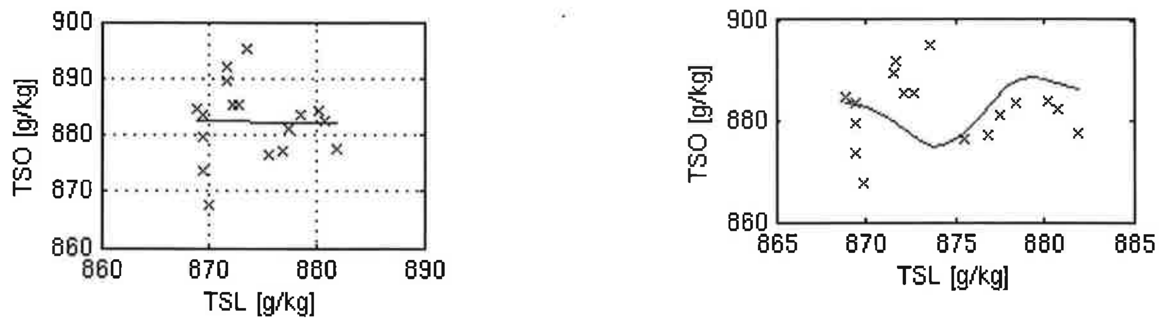
a) Analysis for the year 2005 and the nutrients 'TSL' and 'RA' (corresponding file "tsl_ra2005.dat")



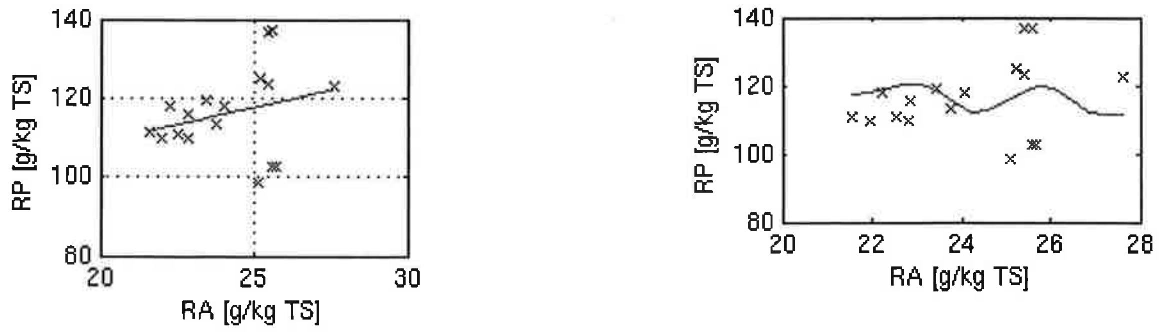
b) Analysis for the year 2008 and the nutrients 'TSL' and 'RA' (corresponding file "tsl_ra2008.dat")



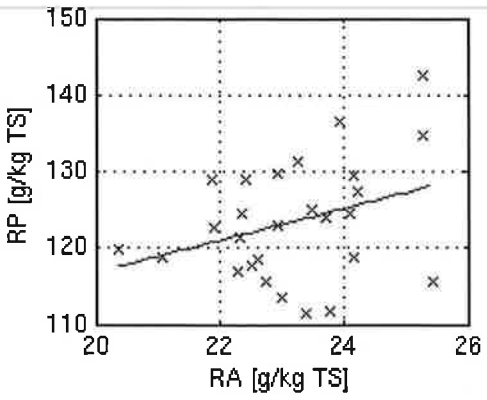
c) Analysis for the year 2005 and the nutrients 'TSO' and 'TSL' (corresponding file "tso_tsl.dat")

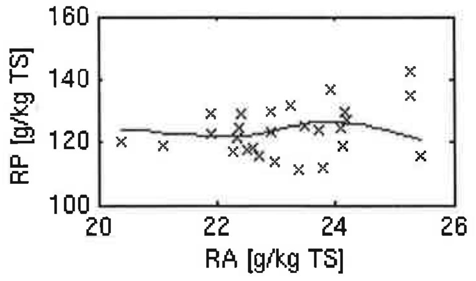


d) Analysis for the year 2005 and the nutrients 'RP' and 'RA' (corresponding file "rp_ra2005.dat")

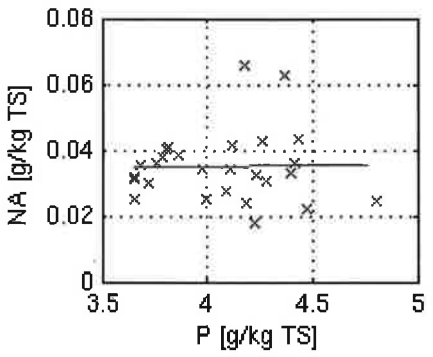


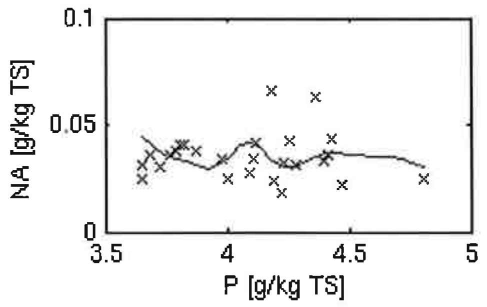
e) Analysis for the year 2008 and the nutrients 'RP' and 'RA' (corresponding file "rp_ra2008.dat")



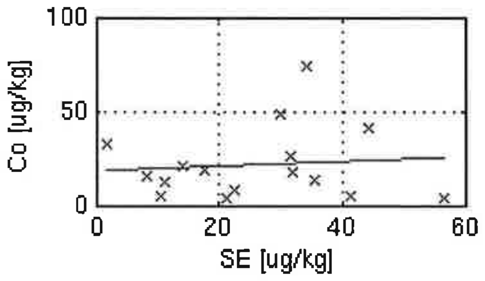


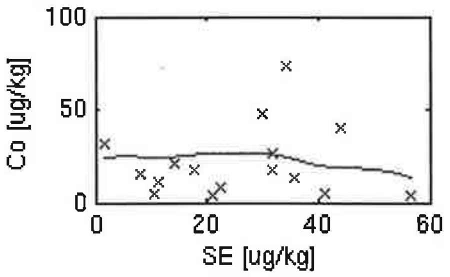
f) Analysis for the year 2008 and the nutrients 'NA' and 'P' (corresponding file "na_p.dat")





g) Analysis for the year 2008 and the nutrients 'Co' and 'SE' (corresponding file "co_se.dat")

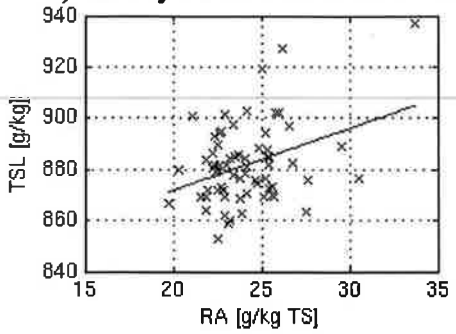


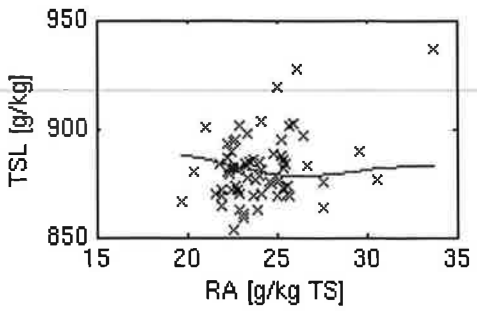


Analysis for the whole given period

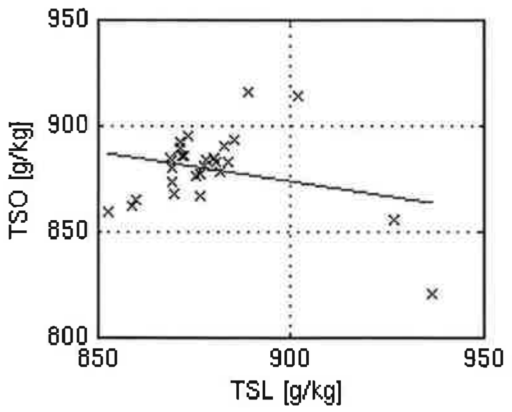
All graphs are for a period of one year. We repeat the calculations for the whole given period. Here we note that the analysis for the nutrients 'Co' and 'SE' is the same as before, since we have no additional useful data outside the year 2008. As before the linear regression is on the left side and the kernel regression computation is shown on the right side.

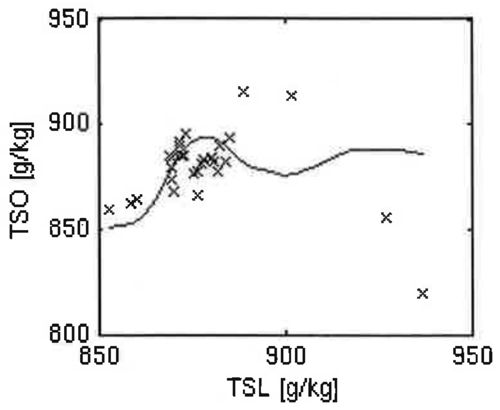
a) Analysis for the nutrients 'TSL' and 'RA' (corresponding file "tsl_ra_whole.dat")



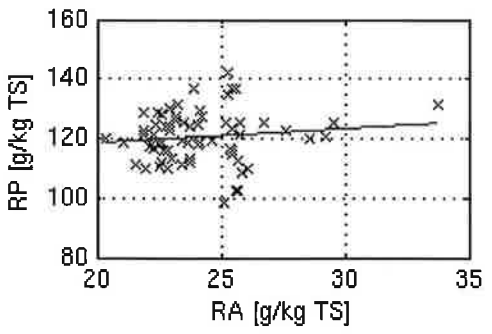


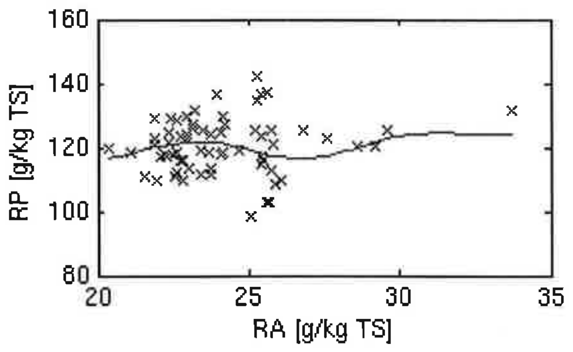
b) Analysis for the nutrients 'TSO' and 'TSL' (corresponding file "tso_tsl_whole.dat")



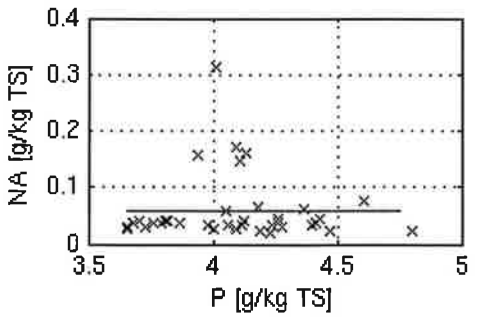


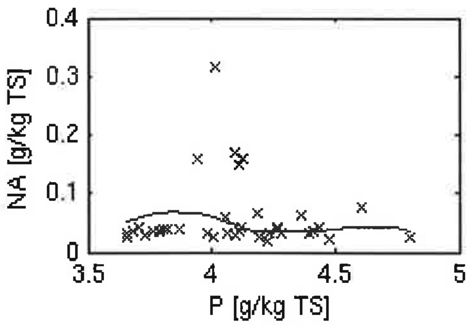
c) Analysis for the nutrients 'RP' and 'RA' (corresponding file "rp_ra_whole.dat")





d) Analysis for the nutrients 'NA' and 'P' (corresponding file "na_p_whole.dat")





Graphical observations and comparison

We see that not all of the assumed linear relationships deliver good results. The linear regression analysis for the year 2005 and the nutrients 'TSL' and 'RA' or the nutrients 'TSO' and 'TSL' are such examples. In fact linear regression is not applicable at all for these. The kernel regression method works better here and thus finds new non-linear correlations between the aforementioned components. A very good example for this is the kernel regression analysis for the year 2005 and the nutrients 'TSO' and 'TSL'.

Furthermore the kernel regression seems to find all correlated components as it does linear regression. But we can only say "seems to find", because the kernel regression works only locally. So outside our data range we can only speculate. The reason, why kernel regression finds more or less correlated components as good as linear regression lies in the non-parametric technique. We don't get linear results, but comparing both methods, we see that the kernel regression is similar and near to the linear results. This is because the kernel regression does not assume any distribution, so it finds nearly linear correlations, too. Playing with the parameters in the kernel regression shows us that we can get very near to linear approximations of our data, if we want to *force* it. This demonstrates the flexibility of the kernel regression method.

On the other hand, kernel regression does not help very much if we can clearly assume a *linear* correlation between the components. An excellent example for this is the comparison between the linear and kernel regression analysis for the years 2005 and 2008 and the nutrients 'RP' and 'RA', where the linear correlation is clearly given. Although the kernel regression also finds a correlation near to the linear regression, the linear method is a better alternative.

We observe that if we analyze the regression methods for the whole given period, the linear regression seems to work better except for the nutrients 'TSL' and 'TSO', where the two pair of values on the right side force the linear regression to deliver a declining line, whereas the kernel regression adjusts the values – as it should – to get a satisfying result.

Numerical observations and comparison

For a numerical comparison we need to calculate an error. The standard deviation is suitable in our case, with which we can compare both methods. The standard deviation is calculated by

$$\sigma=\sqrt{\frac{1}{n}\cdot\sum_{i=1}^{n}\left(f(x_i)-y_i\right)^2}$$

where n is the number of measurements, x_i and y_i are the pair of values and $f(x_i)$ is the value of the regression function at the parameter point x_i. Below is a table with the numerical observations representing the error in each regression method.

Missing values for the nutrient	*Basis for regression is the nutrient*	*Year of analysis*	*Error (linear regression)*	*Error (kernel regression)*
TSL	RA	2005	4.2838	5.3080
TSL	RA	2008	11.0625	10.9749
TSO	TSL	2005	6.5379	8.0942
RP	RA	2005	10.0765	10.7102
RP	RA	2008	7.1560	7.6932
NA	P	2008	0.0105	0.0120
Co	SE	2008	18.5111	18.5752
TSL	RA	Whole given period	14.1645	15.7086
TSO	TSL	Whole given period	17.1067	19.5016
RP	RA	Whole given period	8.4512	8.4627
NA	P	Whole given period	0.0584	0.0592

We are now confronted with the fact that the kernel regression delivers minimal greater errors. In general, we can see that the difference between the kernel and linear method is a small one. We already stated that the linear method works better for the analysis of the whole given period, but why does the kernel regression method deliver a greater error for the nutrients 'TSO' and 'TSL'? The answer lies again in the two pair of values at the right side, where the linear line is nearer than the kernel regression, which affects the error in the kernel method more than in the linear one. As we already noted in the graphical observation, the linear method works clearly better for the nutrients 'RP' and 'RA' in the years 2005 and 2008. The numbers in the table support this fact.

Conclusion

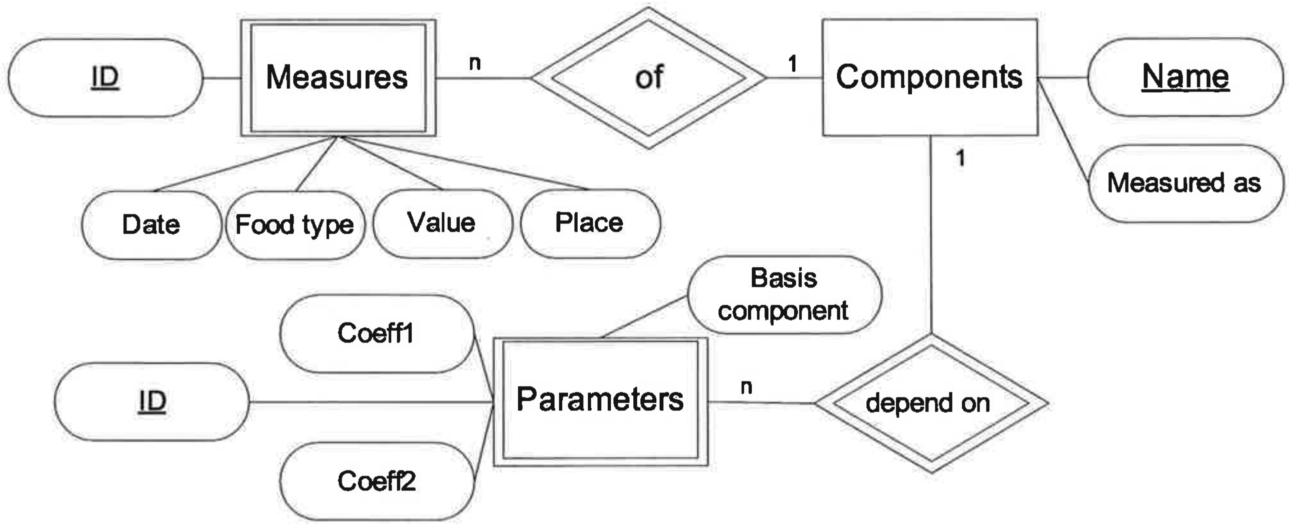
The comparison of the results show us that given a clearly linear correlation, the linear regression method works better. But if the data is somewhat “not very clearly linear”, kernel regression method is sufficient enough. If we regard both methods in the aspects of capability and flexibility, the kernel regression is the better method. In terms of simplicity the linear method is clearly the better one and can be used in many cases. The kernel method however, can up to some degree also cope with linear correlations. So all in all, the more flexible kernel regression method suits our needs better.

Database query execution generating missing values

In this section we assume that correlated nutrients and parameters of linear regression are known and stored in the database. Our question is how such a table will look like when we design a SQL query which computes missing values of one nutrient based on known values of other nutrient and parameters of linear regression and how the SQL query operates on the data.

Database Design

An ER diagram of the feed database together with the parameters could look as in the following figure.



If a component is deleted out of the database, the corresponding measures and the depending parameters have to be deleted too.

SQL schema

We now represent our design in a SQL schema, where the primary keys are underlined and the foreign keys are dotted underlined. Below is a partial representation of our design in a SQL schema.

Components	
Name	***Measured as***
TSO	g/kg
TSL	g/kg
OS	g/kg TS
RA	g/kg TS
RP	g/kg TS
RPK	g/kg TS
RPT	g/kg TS
RPL	g/kg TS
...	...

Measures					
ID	***Food type***	***Place***	***Date***	***Value***	***Name***
1	Wintergerste	Schweiz		870	TSO
2	Sommergerste	Schweiz		870	TSO
3	Wintergerste	Westschweiz			TSO
...	...	...	...	...	...
81	Wintergerste	Schweiz			TSL
82	Sommergerste	Schweiz			TSL
83	Wintergerste	Westschweiz			TSL
84	Gerste	Schweiz	5.6.1992	901.92	TSL
...	...	...	...	...	...

We can see that that the "Measures" table is going to be large, since every component consists of 80 measurement entry fields. Most of the entries for the attributes "Date", "Place" or "Value" will be empty. There are some nutrients like 'OS', 'RPT' or 'RPL', which don't have any measurement values at all, as mentioned before in the section "Database 'Analyses of Barley 1992-2009'".

Parameters				
ID	***Basis component***	***Coeff1***	***Coeff2***	***Name***
1	TSL	1123.7	- 0.2780	TSO
2	RA	108.9586	0.4812	RP
3	RA	823.3514	2.4242	TSL
...	...	...	...	...

In this partial table of our parameters, there are three examples for the parameters of a linear regression. The order of the coefficients in the linear regression matters, so we have to pay attention to this, when we insert new values. We can extract the following informations from the above table:

- Nutrients 'TSO' (missing *y*-values) and 'TSL' (basis for regression) give us for the whole given period the parameters 1123.7 and 0.2780, so

$$y = 1123.7 - 0.2780*x$$

- Nutrients 'RP' (missing *y*-values) and 'RA' (basis for regression) give us for the whole given period the parameters 108.9586 and 0.4812, so

$$y = 108.9586 + 0.4812*x$$

- Nutrients 'TSL' (missing *y*-values) and 'RA' (basis for regression) give us for the whole given period the parameters 823.3514 and 2.4242, so

$$y = 823.3514 + 2.4242*x$$

So if we design a SQL query which has to compute missing values, by this way the information is directly taken out of the feed database and we get the calculated values. The advantage is that if the correlation parameters should be updated or new parameters should be inserted, it can be easily done this way. However, this table represents the design that the parameters are for the whole given period. If we want to restrict the parameters to a limited time period, we can extend the parameters list with entries for the attribute "Basis component" like 'RP_2005' or 'TSL_2005' and fill in the parameters as shown before.

Database queries

For a database query we focus our attention to the linear correlated nutrients 'RP' (missing *y*-values) and 'RA' (basis for regression), so as above

$$y = 108.9586 + 0.4812*x.$$

First we restrict the entries only to the values we need to calculate. Here we have to note that for both nutrients 'RA' and 'RP' there are missing values for the food type named "Gerste undef" and the date '18.05.2004'. So this is the only entry we have to leave out. The SQL query

```
create view RAandRPValues as

select * from
    (select * from Measures where Name = 'RA' or Name = 'RP')
where Date != '18.05.2004'
order by ID;
```

accomplishes this, yielding 158 entries, 79 for each nutrient. The ID numbers for the nutrient 'RA' range from 241 to 320, and for the nutrient 'RP' from 321 to 400.

RAandRPValues

ID	*Food type*	*Place*	*Date*	*Value*	*Name*
241	Wintergerste	Schweiz		26	RA
242	Sommergerste	Schweiz		25	RA
243	Wintergerste	Westschweiz		26.44	RA
...	...	...	...	...	...
321	Wintergerste	Schweiz			RP
322	Sommergerste	Schweiz			RP
323	Wintergerste	Westschweiz			RP
...	...	...	...	...	...

To combine the corresponding values in the calculation and to make the SQL statement easier, we first split up the parts of the table containing the values for the nutrient 'RA' in a temporary view. In the next step, we design a SQL query generating all missing values for the nutrient 'RP' based on the values for 'RA' and the linear regression parameters and saving them in our view "RAandRPvalues".

```
with Parameters p as (
     select Coeff1, Coeff2 from Parameters
     where Basis component = 'RA' and Name = 'RP'
     )
, RAvalues ra as (
     select * from RAandRPValues where ID in (241,320)
     )

update RAandRPValues
set Value = p.Coeff1 + p.Coeff2 * ra.Value
where ID = ra.ID + 80
```

The update clause in the SQL query computes twelve missing values for the nutrient 'RP'. The result of this query is partially illustrated below, where the empty entries of values for the nutrient 'RP' are now filled with the calculated numbers.

RAandRPvalues

ID	*Food type*	*Place*	*Date*	*Value*	*Name*
241	Wintergerste	Schweiz		26	RA
242	Sommergerste	Schweiz		25	RA
243	Wintergerste	Westschweiz		26.44	RA
...	...	...	...	...	...
321	Wintergerste	Schweiz		121.4698	RP
322	Sommergerste	Schweiz		120.9886	RP
323	Wintergerste	Westschweiz		121.6815	RP
324	Gerste	Schweiz	5.6.1992	121.4602	RP
...	...	...	...	...	...

It is advisable to leave the original feed database as it is and to store the values in a new or temporary view, since the correlation parameters can change over time. Another alternative would be to alter the table “Measures” to include one more row called “computed” with a boolean data type or predefined numbers, for example “0” meaning 'measured' and “1” meaning 'computed'. But then again, we have to update all old numbers and pay attention to the entries, which are measured and not computed. Creating a new view and deleting the old, deprecated one is far easier.

Summary

References

[Sauter 10]	Sauter, Stefan Skript „Einführung in die Numerik“, effective 03.06.2010 http://www.math.uzh.ch/index.php?ve_vo_det&key2=1128
[Koláček 09]	KOLÁČEK, Jan. *Kernel Regression Toolbox for Matlab.* 2009. http://www.muni.cz/law/research/publications/862917/
[Teknomo 07]	Teknomo, Kardi (2007) Tutorial on Kernel Regression, effective 26.2.2011 http://people.revoledu.com/kardi/tutorial/Regression/KernelRegression/

Appendix A: Files for the pair of values

File *co_se.dat*		File *na_p.dat*		File *tsl_ra2008.dat*		File *rp_ra2008.dat*	
8.13	15.89	4.3653	0.0630	23.48	885.08	23.48	125.07
14.12	21.01	4.1922	0.0241	21.09	900.64	21.09	118.75
10.46	4.35	4.4343	0.0433	22.93	862.15	22.93	123.01
17.72	18.16	4.8049	0.0244	24.21	881.29	24.21	127.31
34.19	73.61	3.8077	0.0402	20.38	880.13	20.38	119.81
44.11	40.66	3.8699	0.0383	22.29	886.2	22.29	116.9
22.59	8.36	4.2289	0.0180	25.27	894.45	25.27	134.61
56.69	3.51	3.7276	0.0303	23.25	884.15	23.25	131.37
31.93	17.76	4.4740	0.0224	25.45	884.2	25.45	115.64
29.9	48.18	4.1821	0.0657	22.99	881.94	22.99	113.45
31.63	26.45	3.9828	0.0340	22.37	881.99	22.37	124.61
1.77	32.11	3.8202	0.0409	22.33	892.71	22.33	121.26
35.61	13.57	4.1137	0.0343	23.78	876.46	23.78	111.6
11.18	11.74	4.0028	0.0249	21.89	863.89	21.89	129.07
41.21	4.52	4.0951	0.0275	21.91	869.05	21.91	122.72
21.14	3.93	4.2803	0.0309	24.16	902.89	24.16	129.58
		4.1215	0.0418	22.42	878.94	22.42	128.91
		4.3998	0.0329	23.94	862.69	23.94	136.61
		4.4158	0.0363	24.14	870.33	24.14	118.63
		4.2622	0.0426	23.4	897.35	23.4	111.46
		3.6549	0.0316	22.93	869.55	22.93	129.78
		4.2339	0.0324	25.29	887.53	25.29	142.42
		3.7632	0.0360	24.1	884.46	24.1	124.6
		3.6843	0.0357	22.61	894.25	22.61	118.37
		3.7914	0.0376	23.72	885.68	23.72	124.09
		3.6551	0.0315	22.51	889.55	22.51	117.7
		3.6554	0.0251	22.73	895.2	22.73	115.67

File *rp_ra2005.dat*		File *tsl_ra2005.dat*		File *tso_tsl.dat*	
24.05	117.71	24.05	878.47	878.47	883.5
22.85	115.81	22.85	880.75	880.75	882.5
22.24	118.09	22.24	880.25	880.25	884.1
23.42	119.32	23.42	877.51	877.51	881
25.43	123.25	25.43	881.96	881.96	877.5
27.62	122.78	27.62	875.53	875.53	876.4
25.22	125.22	25.22	876.84	876.84	877.2
25.6	102.62	25.6	869.91	869.91	867.7
25.11	98.54	25.11	869.39	869.39	879.6
25.69	102.62	25.69	869.44	869.44	873.5
25.61	136.97	25.61	873.54	873.54	895.2
25.43	136.71	25.43	872.68	872.68	885.4
22.83	109.85	22.83	872.19	872.19	885.4
21.97	109.92	21.97	871.58	871.58	889.6
22.53	110.91	22.53	871.71	871.71	891.9
21.57	111.07	21.57	869.48	869.48	883.7
23.76	113.15	23.76	868.9	868.9	884.7

tso_tsl_whole.dat		*na_p_whole.dat*		*tsl_ra_whole.dat*		*rp_ra_whole.dat*	
927.15	855.6	4.6107	0.0753	25.98	901.92	25.41	114.66
876.70	866.1	4.0554	0.0593	24.72	874.82	25.78	121.13
936.90	819.9	4.2650	0.0388	27.56	863.20	26.09	109.84
859	862.2	4.1095	0.1478	26.16	927.15	25.85	108.36
860.25	864.3	4.0951	0.1709	25.02	919.10	25.72	112.86
852.95	859.2	3.9380	0.1588	30.57	876.70	24.66	119.31
882.65	889.7	4.1291	0.1605	26.54	896.70	29.21	120.59
901.85	913.5	3.7046	0.0421	24.66	875.9	28.62	120.16
889.20	915.5	4.0608	0.0333	19.73	866.50	22.12	117.2
878.47	883.5	4.3653	0.0630	22.97	901.30	23.68	118.59
880.75	882.5	4.1922	0.0241	22.62	881.95	22.97	124.26
880.25	884.1	4.4343	0.0433	24.82	888.55	22.62	112.25
877.51	881.0	4.8049	0.0244	33.73	936.90	33.73	131.6
881.96	877.5	3.8077	0.0402	23.15	859	23.15	127.47
875.53	876.4	3.8699	0.0383	23.18	860.25	23.18	126.13
876.84	877.2	4.2289	0.0180	22.59	852.95	22.59	128.38
869.91	867.7	3.7276	0.0303	26.77	882.65	26.77	125.59
869.39	879.6	4.4740	0.0224	25.77	901.85	25.77	125.46
869.44	873.5	4.1821	0.0657	29.56	889.20	29.56	125.34
873.54	895.2	3.9828	0.0340	24.05	878.47	24.05	117.71
872.68	885.4	3.8202	0.0409	22.85	880.75	22.85	115.81
872.19	885.4	4.1137	0.0343	22.24	880.25	22.24	118.09
871.58	889.6	4.0028	0.0249	23.42	877.51	23.42	119.32
871.71	891.9	4.0951	0.0275	25.43	881.96	25.43	123.25
869.48	883.7	4.2803	0.0309	27.62	875.53	27.62	122.78
868.90	884.7	4.1215	0.0418	25.22	876.84	25.22	125.22
885.28	893.1	4.3998	0.0329	25.6	869.91	25.6	102.62
883.94	882.4	4.4158	0.0363	25.11	869.39	25.11	98.54
		4.2622	0.0426	25.69	869.44	25.69	102.62
		3.6549	0.0316	25.61	873.54	25.61	136.97
		4.2339	0.0324	25.43	872.68	25.43	136.71
		3.7632	0.0360	22.83	872.19	22.83	109.85
		3.6843	0.0357	21.97	871.58	21.97	109.92
		3.7914	0.0376	22.53	871.71	22.53	110.91
		3.6551	0.0315	21.57	869.48	21.57	111.07
		3.6554	0.0251	23.76	868.90	23.76	113.15
		4.0112	0.3149	25.4	885.28	25.4	116.91
				21.88	883.94	21.88	121.05
				23.48	885.08	23.48	125.07
				21.09	900.64	21.09	118.75
				22.93	862.15	22.93	123.01
				24.21	881.29	24.21	127.31
				20.38	880.13	20.38	119.81
				22.29	886.2	22.29	116.9
				25.27	894.45	25.27	134.61
				23.25	884.15	23.25	131.37
				25.45	884.2	25.45	115.64
				22.99	881.94	22.99	113.45
				22.37	881.99	22.37	124.61
				22.33	892.71	22.33	121.26

		23.78 876.46	23.78 111.6
		21.89 863.89	21.89 129.07
		21.91 869.05	21.91 122.72
		24.16 902.89	24.16 129.58
		22.42 878.94	22.42 128.91
		23.94 862.69	23.94 136.61
		24.14 870.33	24.14 118.63
		23.4 897.35	23.4 111.46
		22.93 869.55	22.93 129.78
		25.29 887.53	25.29 142.42
		24.1 884.46	24.1 124.6
		22.61 894.25	22.61 118.37
		23.72 885.68	23.72 124.09
		22.51 889.55	22.51 117.7
		22.73 895.2	22.73 115.67
		22.76 873.4	22.76 123.71

Appendix B: Own Matlab-Programs

File qrsolve_er.m

```
function [x] = qrsolve_er(X,f)

clc
[n k] = size(X);
M = zeros(n,k+1);
M(1:n,1) = 1;
M(:,2:k+1) = X(:,1:k);
A=M;
R = A'*A;
x = qrsolve(R,A'*f);
x
```

File qrsolve.m

```
function x = qrsolve(A,b);

R = A;
n = size(R);
QH = eye(n);

for i = 1:(n-1)

    a = R(i:n,i);
    u = a;
    u(1) = a(1)/abs(a(1))*(abs(a(1))+norm(a));
    beta = 1/(norm(a)*(abs(a(1))+norm(a)));
    I = eye(n+1-i);
    P = I-beta*u*u';
    P = blkdiag(eye(i-1),P);
    R = P*R;                        % Obere rechte Dreiecksmatrix
    QH = P*QH;                      % QH =P=P(n-1) · P(n-2) · ... ·P1

end

y = QH*b;

for i = n:-1:1;

    s = y(i,1);

    for j = i+1:n;

        s = s-R(i,j)*x(j,1);

    end

        x(i,1) = s/R(i,i);

end

x
```