Statistical Methods:
Prediction of Soil Parameters through Near Infrared Spectroscopy

Kazimir Menzel
Markus Pawellek

August 16, 2016

Abstract

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Contents

1 Introduction **1**

2 Background **1**
2.1 Soil Parameters 1
2.2 Near Infrared Spectroscopy 1

3 Methodology **2**
3.1 Measured Data 2
3.2 Statistical Model 2
3.3 Multivariate Linear Regression 3
3.4 Mallows' Cp 3
3.5 Simulated Annealing 3
3.6 Model Validation 4

4 Model Selection **4**
4.1 Choosing a Neighbor 4
4.2 Calibration 4
4.3 Goodness of Fit 4

5 Simulation **4**

6 Conclusion **4**

A Prediction Parameters **i**

B R Source Code **iii**

Statistical Methods:
Prediction of Soil Parameters through Near Infrared Spectroscopy

Kazimir Menzel
kazimir.menzel@me.com

Markus Pawellek
markuspawellek@gmail.com

Abstract

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

1 Introduction

Through soil analyses or soil testing one can get certain chemical and physical information about the used soil like the concentration of soil organic carbon (SOC) or the pH-value. With these Measurements, known as soil parameters, it is possible to assist in solving soil-related problems such as optimizing plant growth.

However the direct measurement of soil parameters is very costly and error-prone. Therefore methods for fast and cheap determination of these parameters play a fundamental and vital role in particular fields like agriculture, geochemistry and ecology.

Albeit developed in the 1960s, Near Infrared Spectroscopy (NIRS) only gained traction during the 1990s as sufficient computing power became increasingly available. NIRS provides a cheap alternative to other methods for the analysis of soil composition .

After a short summary of the physical background of NIRS, we will lay out some of the difficulties in applying NIRS to real world problems. The rest of the paper is dedicated to a methodological framework to address some of those obstacles. In particular we propose a computationally efficient way to identify optimal parameters on a training sample with known data to be used for further analysis of NIRS on soil samples.

2 Background

2.1 Soil Parameters

Let A be any substance in a given soil sample with volume $V \in (0, \infty)$ and let $n_A \in (0, \infty)$ be the amount of A in the sample. Then the molar concentration c_A is given by

$$c_A \coloneqq \frac{n_A}{V}$$

Now let c_0 be the molar concentration of this whole sample. We define the amount-of-substance fraction (ASF) p_A of A as

$$p^{(A)} \coloneqq 100\% \cdot \frac{c_A}{c_0} = \frac{n_A}{n_0}$$

In this report, we concentrate on three important soil parameters that are given by existing NIRS-measurements. The first two parameters $p^{(\mathrm{SOC})}$ and $p^{(\mathrm{N})}$ are the ASFs relating to soil organic carbon (SOC) and nitrogen in a given soil sample. SOC refers to the carbon in the sample that is bound in an organic compound. The third parameter is the pH-value that specifies the acidity of an aqueous solution. It links to the concentration of hydronium ions $c_{\mathrm{H_3O^+}}$.

$$\mathrm{pH} \coloneqq -\lg c_{\mathrm{H_3O^+}} = -\frac{\ln c_{\mathrm{H_3O^+}}}{\ln 10}$$

2.2 Near Infrared Spectroscopy

NIRS uses electromagnetic waves,famously known as light, with wavelengths ranging from $780\,\mathrm{nm}$ to $3000\,\mathrm{nm}$, the so called near infrared spectrum. This area is called the near infrared region and is the most energetic one of the infrared light.

An emitted light wave with wavelength $\lambda \in (0, \infty)$ interacts with a soil sample in three ways: It can be reflected, absorbed or transmitted. For most soil samples measuring the transmittance of light waves is not sensible as the thickness of these samples varies. Since the absorptance cannot be determined directly, reflectance is used.

Reflection can be split in specular and diffuse reflection. NIRS uses the latter as it penetrates the sample the most. As a consequence, diffuse reflected light is hemispherically scattered and contains information about the soil sample composition. For a more detailed view on this topic please refer to [Tutorial].

The reflectance

$$\varrho\colon (0,\infty) \to (0,\infty), \qquad \varrho(\lambda) \coloneqq \frac{P_\mathrm{r}(\lambda)}{P_0}$$

of a surface depending on wavelength λ of a light wave is given by the amount of radiation power $P_\mathrm{r}(\lambda)$ that is reflected from the surface divided by the initial power P_0 of the light wave. In our case, this function ϱ is defined as the near infrared spectrum of the soil sample.

Absorptance itself originates from the existence of vibrational modes in molecules. A photon with a wavelength $\lambda \in (0,\infty)$ can only be absorbed if the appropriate frequency f exactly matches a multiple of the transition energy of the bond or group that vibrates.This is why the spectra of soil samples are formed of overtones and combination bands.

Due to the similarity of diffuse reflected and transmitted light we can use Beer-Lambert's law as a relation of the attenuation of light and the properties of samples. Let $n \in \mathbb{N}$ be the count of different substances in a sample and c_i be the molar concentration of the ith substance for $i \in \mathbb{N}, i \leq n$. Is again $\lambda \in (0,\infty)$ the wavelength of the used light then there exist certain coefficients $\varepsilon_i(\lambda)$ for all $i \in \mathbb{N}, i \leq n$ such that

$$-\ln \varrho(\lambda) = \sum_{i=1}^{n} \varepsilon_i(\lambda) c_i$$

3 Methodology

3.1 Measured Data

As a single spectrum contains overlapping information, it is necessary to determine both relevant wavelengths and the respective parameters to apply NIRS to practical problems. To select wavelengths and determine parameters we use an example data set, which contains $p^{(\mathrm{SOC})}, p^{(\mathrm{N})}$, pH and wave reflectances of 319 wavelengths ranging from $1400\,\mathrm{nm}$ to $2672\,\mathrm{nm}$ by steps of $4\,\mathrm{nm}$ for 533 samples.

We define Λ as the set of all measured wavelengths. The reflectance $\varrho(\lambda)$ of a sample at a wavelength $\lambda \in \Lambda$ is recorded as

$$-\lg \varrho(\lambda) = -\frac{\ln \varrho(\lambda)}{\ln 10}$$

Figure 1 shows six randomly chosen soil spectra in a diagram.

3.2 Statistical Model

Let $n \in \mathbb{N}$ be the size of the data set and $k \in \mathbb{N}$ with $k \leq n$ the number of measured wavelengths. We define ϱ_i as the soil spectrum of the ith sample for every $i \in \mathbb{N}, i \leq n$. λ_j is the jth measured wavelength for every $j \in \mathbb{N}, j \leq k$. We will alternatively refer to these as predictors.Then according to section 3.1 the measured reflectance values are x_{ij} with

$$x_{ij} \coloneqq -\lg \varrho_i(\lambda_j)$$

for every $i, j \in \mathbb{N}, i \leq n, j \leq k$.

We define the measured ASF of SOC of the ith sample for every $i \in \mathbb{N}, i \leq n$ as $p_i^{(\mathrm{SOC})}$ to which we will also refer to as response variable. To simplify notation, we then define the n-dimensional vector

$$p^{(\mathrm{SOC})} \coloneqq \left(p_i^{(\mathrm{SOC})}\right)$$

The Beer-Lambert law allows us to make assumptions on the relations between soil spectra and the response variable. We saw in section 2.2 that the logarithmised reflectance can be written as a linear combination of molar concentrations. Hence, it seems plausible to assume that an ASF can be represented by a linear combination of logarithmised reflectance values.

Now let $P^{(\mathrm{SOC})}$ be the appropriate random vector to $p^{(\mathrm{SOC})}$. Then under the above assumption the expected values are given for all $i \in \mathbb{N}, i \leq n$ by

$$\mathbb{E}\, P_i^{(\mathrm{SOC})} \coloneqq \beta_0^{(\mathrm{SOC})} + \sum_{j=1}^{k} x_{ij} \beta_j^{(\mathrm{SOC})}$$

which simplifies with an $\mathbb{X} \in \mathbb{R}^{n\times(k+1)}$, called design matrix, and a parameter vector $\beta^{(\mathrm{SOC})} \in \mathbb{R}^{k+1}$ to

$$\mathbb{E}\, P^{(\mathrm{SOC})} = \mathbb{X}\beta^{(\mathrm{SOC})}$$

To capture the stochastic part of $P^{(\mathrm{SOC})}$, we extend the model to

$$\begin{aligned} P^{(\mathrm{SOC})} &= \mathbb{X}\beta^{(\mathrm{SOC})} + \varepsilon^{(\mathrm{SOC})} \\ \mathbb{E}\,\varepsilon^{(\mathrm{SOC})} &= 0, \qquad \operatorname{cov} \varepsilon^{(\mathrm{SOC})} = (\sigma^2)^{(\mathrm{SOC})}\mathrm{I} \end{aligned}$$

where $(\sigma^2)^{(\mathrm{SOC})} \in (0,\infty)$. Following common practice in physics and chemistry, we further assume that

$$\varepsilon^{(\mathrm{SOC})} \sim \mathcal{N}\left(0, (\sigma^2)^{(\mathrm{SOC})}\mathrm{I}\right)$$

This results in the complete model

$$P^{(\mathrm{SOC})} \sim \mathcal{N}\left(\mathbb{X}\beta^{(\mathrm{SOC})}, (\sigma^2)^{(\mathrm{SOC})}\mathrm{I}\right)$$

The model for the second response variable $P^{(\mathrm{N})}$ is constructed in analogy.

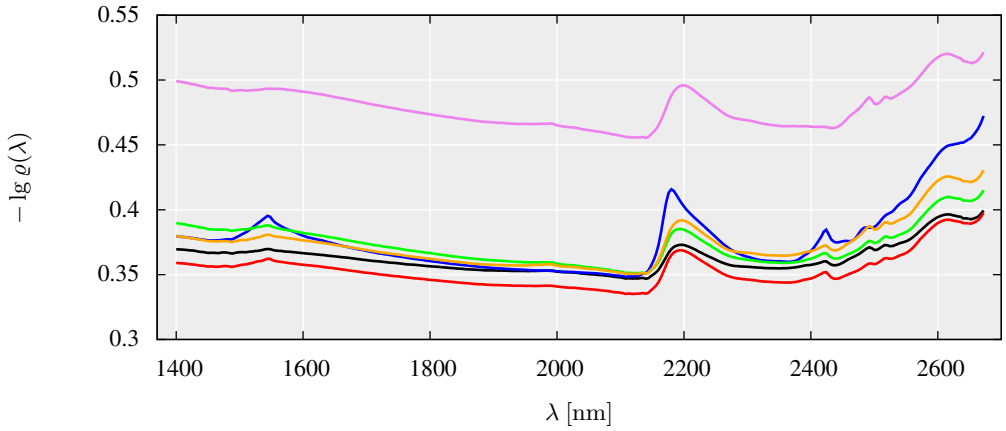


Figure 1: This figure shows near infrared soil spectra of six randomly chosen soil samples obtained from the used dataframe.

The case for the pH is slightly different, though. When modelling the corresponding random variable we have to adjust the model as the pH is a logarithmised molar concentration. We therefore have to include this into the expected value of the corresponding random variables

$$\mathbb{E}\,\overline{\mathrm{pH}}_i := \beta_0^{(\mathrm{pH})} + \sum_{j=1}^{k} \ln(x_{ij})\beta_j^{(\mathrm{pH})}$$

and denote the corresponding matrix by $\mathbb{X}_{\ln}$.

3.3 Multivariate Linear Regression

Multiple linear regression (MLR) or multivariate linear regression is a statistical method for estimating parameters of linear relations between a response variable and a set of predictors and to use these to predict new responses. Let $\mathbb{X} \in \mathbb{R}^{n\times(k+1)}, n, k \in \mathbb{N}, k < n$ be the design matrix, $\sigma^2 \in (0,\infty)$ and Y be the random vector variables with

$$Y \sim \mathcal{N}\left(\mathbb{X}\beta, \sigma^2 \mathrm{I}_n\right)$$

for a certain $\beta \in \mathbb{R}^{k+1}$ Then through the maximum-likelihood-method and a correction we get two best unbiased estimators $\hat{\beta}, \hat{\sigma}^2$ for β and σ^2

$$\hat{\beta}(Y) = \left(\mathbb{X}^{\mathrm{T}}\mathbb{X}\right)^{-1}\mathbb{X}^{\mathrm{T}}Y$$
$$\hat{\sigma}^2(Y) = \frac{1}{n-(k+1)}\left\|Y - \mathbb{X}\hat{\beta}(Y)\right\|^2$$

Now let $y := (y_i) \in \mathbb{R}^n$ be a realization of Y. Then we define

$$\hat{y} := \mathbb{X}\hat{\beta}(y) = \mathbb{X}\left(\mathbb{X}^{\mathrm{T}}\mathbb{X}\right)^{-1}\mathbb{X}^{\mathrm{T}}y$$
$$\hat{\sigma}^2 := \hat{\sigma}^2(y)$$

3.4 Mallows' Cp

At this point, the model is specified using $k+1 = 320$ predictors for each response variable, using the whole measured spectra for each soil sample. [this is wrong] We know from 2.2 that the light waves contain redundant information. This might lead to overfitting, i.e. the variance of our estimated parameters $\hat{\beta}_i(Y)$ might be too large, compromising their usability for future measurements. To address this problem, it is sensible to limit each actual model to a "good" subset of the predictors. Hence, our task becomes to select the best or at least a "sufficiently" good model M defined by

$$M \subset \Lambda \cup \{\lambda_0\}$$

where λ_0 stands for the intercept. We denote the design matrix for each M by $\mathbb{X}^{(M)}$. Applying MLR to the new design matrix yields the new estimators

$$\hat{\beta}^{(M)}(Y) = \left(\mathbb{X}^{(M)\mathrm{T}}\mathbb{X}^{(M)}\right)^{-1}\mathbb{X}^{(M)\mathrm{T}}Y$$
$$\left(\hat{\sigma}^2\right)^{(M)}(Y) = \frac{1}{n-|M|}\left\|Y - \mathbb{X}^{(M)}\hat{\beta}^{(M)}(Y)\right\|^2$$

To determine the "goodness" of M, we use Mallow's Cp given by

$$\mathrm{Cp}^{(M)} := \frac{1}{\hat{\sigma}^2}\sum_{i=1}^{n}\left(y_i - \hat{y}_i^{(M)}\right)^2 - n + 2\,|M|$$

The minimization this value is equivalent to the minimization of the sum of predicted squared errors (SPSE).

3.5 Simulated Annealing

The set of predictors contains $k = 319$ free selectable elements (the constant shall remain). Therefore the power set $\mathcal{P}$, the set of possible subsets M, contains of $2^k = 2^{319}$ elements. If we want to find a subset M so that for all $N \in \mathcal{P}$

the inequation holds

$$\mathrm{Cp}^{(M)} \leq \mathrm{Cp}^{(N)}$$

we have to calculate $\mathrm{Cp}^{(N)}$ for every $N \in \mathcal{P}$. But this is a calculation beyond our current computing power.

Simulated annealing (SA) is a probabilistic technique for approximating the global optimum of a given function. Specifically, it is a metaheuristic to approximate global optimization in a large search space. It simulates the slow cooling of a thermodynamic system through random numbers. With this algorithm it is possible to find a "good" local minimum in a short time.

The algorithm works on an arbitrary set, in our case $\mathcal{P}$. Let $x_0 \in \mathcal{P}$ be the initial set of predictors, $T_0 \in (0, \infty)$ be the initial temperature of the system and $i_{\max} \in \mathbb{N}$ be the maximal number of time steps. Then the algorithm needs certain functions.

- $\mathrm{cost}\colon \mathcal{P} \to \mathbb{R}$
  Calculates the cost of a given predictor set.
- $\mathrm{temp}\colon \mathbb{R} \times \mathbb{N}^2 \to (0, \infty)$
  Calculates the temperature according to the given initial temperature and time steps. It is a monotonically decreasing function in the second parameter.
- $\mathrm{nbr}\colon \mathcal{P} \to \mathcal{P}$
  Generates a random neighbor of a given predictor set.
- $\mathrm{prob}\colon \mathbb{R}^2 \times (0, \infty) \to [0, 1]$
  Calculates the probability of changing the current set or state to the neighbor.
- $\mathrm{rnd}(0, 1)$
  Returns a random number in the interval $[0, 1]$.

The listing shows one variant of the pseudocode of the SA algorithm.

Listing: SA algorithm

$c_0 = \mathrm{cost}(x_0)$

for $(i = 1, i \leq i_{\max})$ {
  $T = \mathrm{temp}(T_0, i, i_{\max})$

  $x_1 = \mathrm{nbr}(x_0)$
  $c_1 = \mathrm{cost}(x_1)$

  if $(\mathrm{prob}(c_0, c_1, T) \geq \mathrm{rnd}(0, 1))$ {
    $x_0 = x_1$
    $c_0 = c_1$
  }
}

3.6 Model Validation

4 Model Selection

4.1 Choosing a Neighbor

As said in section 3.5 with simulated annealing we want to select a good model for the prediction. For this we have to define the functions and parameters of the algorithm. The most important one is the nbr-function whose purpose it is to choose a neighbor with high quality since the final solution will be determined by a sequence of neighbors. In the most cases it is best to select a neighbor not too far away from the given subset.

Our nbr-function generates a random natural number $r \in \{2, \ldots, k+1\}$. This number represents the index of a measured wavelength. If this predictor is already in our current subset then we remove it. In the other case we append it to the subset. That way our function obviously picks a near neighbor. The pseudocode is shown in the following listing.

Listing: nbr-function

function $\mathrm{nbr}(M)${
  $r = \mathrm{rnd}\{2, \ldots, k+1\}$

  if $(\lambda_r \in M)${
    $\tilde{M} = M \setminus \{\lambda_r\}$
  }**else**{
    $\tilde{M} = M \cup \{\lambda_r\}$
  }

  return $\tilde{M}$
}

All other functions were defined with a standard scheme, respectively. It is clear that we set

$$\mathrm{cost}(M) \coloneqq \mathrm{Cp}^{(M)}$$

In most applications prob is defined to be an analogy with the transitions of a physical system.

$$\mathrm{prob}(c_0, c_1, T) \coloneqq \exp\left(\frac{c_0 - c_1}{T}\right)$$

Details of temp are not really important as long as it monotonically decreases in the second parameter. So let $\alpha \in (0, 1)$.

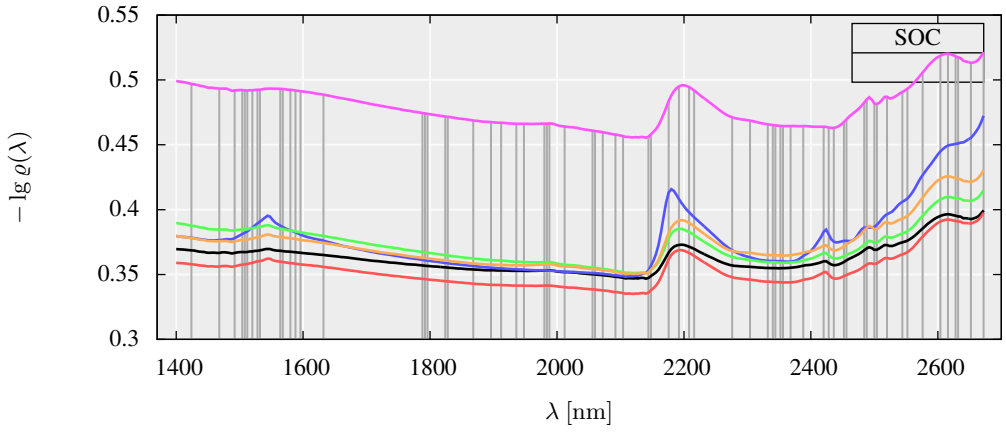
$$\mathrm{temp}(T_0, i, i_{\max}) \coloneqq T_0 \alpha^i$$

4.2 Calibration

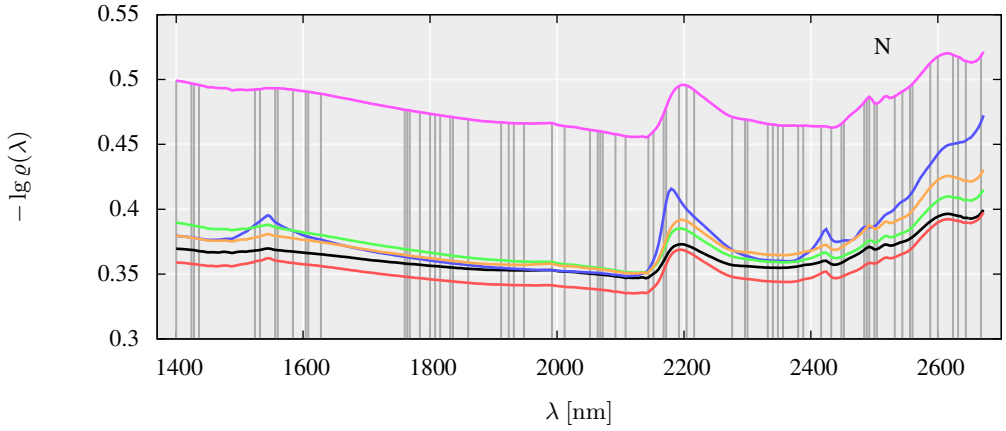
4.3 Goodness of Fit

5 Simulation

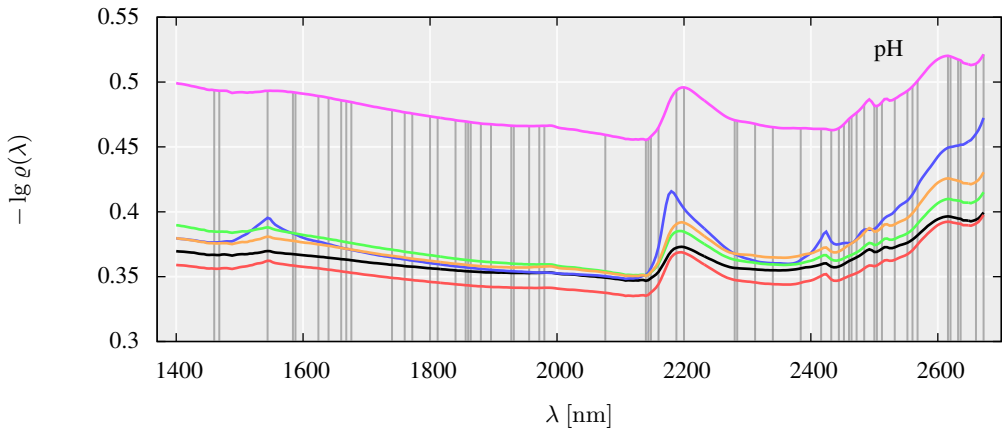
6 Conclusion


SOC
0.55
0.5
0.45
0.4
0.35
0.3
1400
1600
1800
2000
2200
2400
2600
λ [nm]
− lg ϱ(λ)

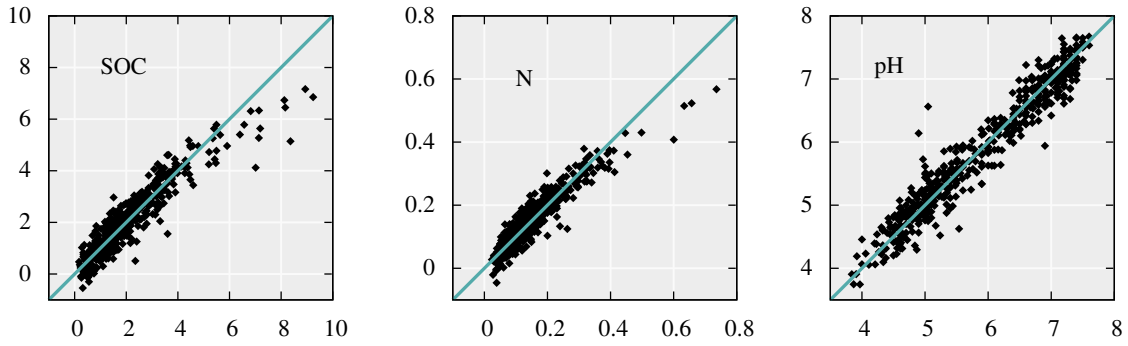
(a) SOC


N
0.55
0.5
0.45
0.4
0.35
0.3
1400
1600
1800
2000
2200
2400
2600
λ [nm]
− lg ϱ(λ)

(b) N


pH
0.55
0.5
0.45
0.4
0.35
0.3
1400
1600
1800
2000
2200
2400
2600
λ [nm]
− lg ϱ(λ)

(c) pH



(d) SOC (e) N (f) pH

Figure 2: correlation

References

A Prediction Parameters

Table 1: parameter of SOC

λ_i [nm]	$\beta_i^{(\mathrm{SOC})}$	λ_i [nm]	$\beta_i^{(\mathrm{SOC})}$	λ_i [nm]	$\beta_i^{(\mathrm{SOC})}$	λ_i [nm]	$\beta_i^{(\mathrm{SOC})}$
—	-1.37	1696	-1166.38	2108	-1382.98	2460	-361.83
1400	826.04	1744	1424.94	2120	1645.94	2484	2332.83
1424	-1324.59	1788	-2087.12	2144	3204.64	2488	-2705.44
1436	-642.72	1792	1898.5	2148	-2621.19	2496	1771.59
1464	-980.9	1800	-1558.26	2180	-613.88	2508	-4389.54
1468	1201.95	1808	1438.45	2192	1799.45	2512	3378.91
1516	1117.94	1832	2504.93	2204	-2726.43	2516	-2699.09
1536	2049.48	1836	-1326.38	2216	1810.36	2520	2754.49
1540	-2041.89	1856	-2577.96	2240	-1500.87	2548	-2761.22
1544	1154.3	1880	1275.66	2248	1717.8	2552	2224.56
1552	-1298.43	1912	-2522.78	2272	-1654.19	2580	1071.07
1556	2252.06	1932	1325.29	2304	1191.13	2600	-1589.8
1560	-2277.97	1948	1592.08	2332	-2318.41	2620	1163.85
1580	-950.58	1980	-1994.92	2340	3079.92	2628	1414.16
1592	-1627.88	1984	3827.51	2348	-1970.47	2632	-2540.03
1596	1697.58	1988	-3669.48	2352	-2222.55	2636	1154.67
1612	857.87	2012	1860.69	2356	3881.99	2648	-1231.84
1624	1472.47	2052	-2273.32	2420	-1023.57	2672	771.23
1628	-2702.99	2072	3197.8	2432	2104.87		
1632	3339.95	2092	-1376.49	2436	-2772.6		
1636	-1430.86	2100	-1309.64	2440	1443.54		

Table 2: parameter of SOC

λ_i [nm]	$\beta_i^{(\mathrm{N})}$	λ_i [nm]	$\beta_i^{(\mathrm{N})}$	λ_i [nm]	$\beta_i^{(\mathrm{N})}$	λ_i [nm]	$\beta_i^{(\mathrm{N})}$
—	-0.03	1824	-322.95	2176	-55.01	2480	146.55
1432	57.15	1828	264.85	2188	68.57	2484	-124.23
1436	-96.58	1860	-89.78	2208	-198.14	2500	223.14
1452	-49.73	1904	91.91	2216	182.95	2504	-279.93
1468	48.52	1912	-207.41	2276	-269.65	2520	55.77
1476	-83.09	1924	-117.17	2280	138.84	2536	93.88
1480	79.57	1932	120.37	2300	125.9	2540	-101.02
1520	76.62	1948	138.91	2332	-140.59	2548	-226.8
1536	41.5	1988	-96.24	2336	135.15	2552	202.85
1556	89.72	2012	123.7	2344	182.6	2576	79.15
1560	-109.73	2052	-193.08	2348	-315.73	2596	-50.8
1580	-103.8	2060	125.31	2356	175.07	2604	-67.93
1608	73.7	2108	-191.7	2420	-71.9	2620	86.74
1668	-71.2	2136	92.38	2432	139.37	2644	-82.74
1772	99.88	2144	190.85	2436	-139.42	2672	57.07
1780	-76.49	2152	-176.58	2452	89.38		
1820	222.91	2164	58.55	2456	-69.74		

Table 3: parameter of SOC

λ_i [nm]	$\beta_i^{(\mathrm{pH})}$	λ_i [nm]	$\beta_i^{(\mathrm{pH})}$	λ_i [nm]	$\beta_i^{(\mathrm{pH})}$	λ_i [nm]	$\beta_i^{(\mathrm{pH})}$
—	5.85	1740	-251.79	2140	427.84	2464	-749.9
1452	201.07	1772	274.75	2144	-462.86	2468	482.33
1456	-235.73	1800	329.93	2148	332.95	2472	-345.54
1472	147.99	1840	218.82	2164	-85.81	2480	75.03
1520	-142.72	1856	-354.15	2180	81.84	2504	162.03
1572	166.6	1860	694.27	2212	-272.38	2516	-145.78
1576	-214.97	1864	-716.34	2228	300.92	2528	-180.23
1584	419.67	1892	361.97	2280	452.12	2552	-185.8
1588	-213.07	1896	-1030.42	2284	-634.41	2564	372.02
1628	398.62	1900	718.02	2312	249.04	2588	223.49
1632	-429.17	1904	-808.16	2340	-340.18	2612	-355.3
1636	336.6	1908	312.91	2376	259.71	2628	252.09
1640	-329.38	1928	686.48	2420	-124.78	2640	-312.24
1660	-191.75	1932	-419.56	2428	187.87	2660	-114.89
1668	341.29	1972	-266.46	2444	-604.36	2668	253.54
1672	-211.02	1980	462.93	2448	405.98		
1680	-186.77	2076	-283.23	2460	607.91		

B R Source Code

Listing: utils.r

```
# calculate the gram matrix of a given matrix
# gram.mat <- function(mat){
#   #return
#   t(mat) %*% mat
# }

# mlr.transf.obs.vec <- function(obs_vec, design_mat){

# }

# get matrix for calculating parameters
# mlr.par.mat = function(design_mat){
#   transp_design_mat <- t(design_mat)

#   # return
#   solve(transp_design_mat %*% design_mat) %*% transp_design_mat
# }

# calculate parameters
# mlr.par = function(obs_vec, design_mat){
#   # return
#   as.vector(mlr.par.mat(design_mat) %*% obs_vec)
# }

# initialize observables and design matrix (needed for fast calculation)
init.data = function(obs_vec, design_mat){
    gv_obs_vec <<- obs_vec
    gv_design_mat <<- design_mat
    gv_sample_size <<- length(gv_obs_vec)
    gv_par_size <<- dim(gv_design_mat)[2]

    # preprocessing
    gv_gram_design_mat <<- t(gv_design_mat) %*% gv_design_mat
    gv_transf_obs_vec <<- t(gv_design_mat) %*% gv_obs_vec
}

# generate pseudo observables (needs init.data; first use another function to calculate global
    variables: gv_expect_vec, gv_sd)
# gen.pseudo.obs.vec = function(){
#   # return
#   rnorm(gv_sample_size, gv_expect_vec, gv_sd)
# }

# initialize global variables for given observables and design matrix (needed for fast calculation
     of multiple linear regression and model selection)
mlr.init = function(){
    # transp_design_mat <- t(design_mat)

    # global variables
    # gv_mlr_design_mat <<- design_mat
    # gv_mlr_par_size <<- dim(design_mat)[2]
    # gv_mlr_gram_design_mat <<- transp_design_mat %*% design_mat

    # gv_mlr_obs_vec <<- obs_vec
    # gv_mlr_sample_size <<- length(obs_vec)
    # gv_mlr_transf_obs_vec <<- transp_design_mat %*% obs_vec
    # gv_mlr_expect_vec <<- gv_design_mat %*% gv_mlr_par_vec

    gv_mlr_par_vec <<- solve(gv_gram_design_mat, gv_transf_obs_vec)
    res_vec <- gv_obs_vec - (gv_design_mat %*% gv_mlr_par_vec)
    gv_mlr_var <<- ( as.numeric( t(res_vec)%*%res_vec) ) / (gv_sample_size - gv_par_size)
    gv_mlr_inv_var <<- 1.0 / gv_mlr_var
```

```
    # gv_mlr_var <<- gv_mlr_rss  / (gv_mlr_sample_size - gv_mlr_par_size)
}

#
ms.init.dist = function(idx_vec){
    par_vec <- solve(gv_gram_design_mat[idx_vec,idx_vec], gv_transf_obs_vec[idx_vec])

    # global: model selection expectation vector
    gv_expect_vec <<- as.matrix(gv_design_mat[,idx_vec]) %*% par_vec

    res_vec <- gv_obs_vec - gv_expect_vec

    # global: model selection standard deviation
    gv_sd <<- sqrt( as.numeric( t(res_vec) %*% res_vec ) / (gv_sample_size - length(idx_vec)) )
}

# initialize new observable with the same length (needs mlr.init)
# mlr.init.obs.vec = function(obs_vec){
#   gv_mlr_obs_vec <<- obs_vec
#   # gv_mlr_sample_size <<- length(obs_vec)
#   gv_mlr_transf_obs_vec <<- t(gv_mlr_design_mat) %*% obs_vec
#   gv_mlr_par_vec <<- solve(gv_mlr_gram_design_mat, gv_mlr_transf_obs_vec)
#   gv_mlr_expect_vec <<- gv_mlr_design_mat %*% gv_mlr_par_vec
#   gv_mlr_res_vec <<- obs_vec - gv_mlr_expect_vec
#   gv_mlr_rss <<- as.numeric(t(gv_mlr_res_vec)%*%gv_mlr_res_vec)
#   gv_mlr_var <<- gv_mlr_rss  / (gv_mlr_sample_size - gv_mlr_par_size)
#   gv_mlr_inv_var <<- 1.0 / gv_mlr_var
# }

# mlr.init.design.mat = function(design_mat){

# }

# get hat-matrix of a given design-matrix
# design_mat must have full rank
# mlr.hat.mat = function(design_mat){
#   # return
#   design_mat %*% mlr.par.mat(design_mat)
# }

# multiple linear regression residual sum of squares (rss)
# mlr.rss = function(obs_vec, design_mat){
#   hat_mat <- mlr.hat.mat(design_mat)
#   res <- obs_vec - (hat_mat %*% obs_vec)

#   # return
#   as.numeric(t(res) %*% res)
# }

# multiple linear regression variance estimator
# mlr.var = function(obs_vec, design_mat){
#   # return
#   (mlr.rss(obs_vec, design_mat)) / (length(obs_vec) - dim(design_mat)[2])
# }

ms.expect.vec = function(idx_vec){
    par_vec <- solve(gv_gram_design_mat[idx_vec,idx_vec], gv_transf_obs_vec[idx_vec])

    # return
    as.matrix(gv_design_mat[,idx_vec]) %*% par_vec
}

# get residual sum of squares for given model (needs mlr.init)
ms.rss = function(idx_vec){
    par_vec <- solve(gv_gram_design_mat[idx_vec,idx_vec], gv_transf_obs_vec[idx_vec])
    res_vec <- gv_obs_vec - ( as.matrix(gv_design_mat[,idx_vec]) %*% par_vec )
```

```
    # return
    as.numeric( t(res_vec) %*% res_vec )
}

ms.spse = function(idx_vec){
    # return
    ms.rss(idx_vec) + (2 * gv_mlr_var * length(idx_vec))
}

# get mallows cp for certain model
ms.cp = function(idx_vec){
    # return
    (ms.rss(idx_vec) * gv_mlr_inv_var) + (2*length(idx_vec)) - gv_sample_size
}

# model selection: forward selection method
# ms.fwd.sel = function(obs_vec, design_mat, invgv_mlr_var){
#   full_idx_vec <- seq(1, dim(design_mat)[2])
#   # first column will be used every time
#   idx_vec <- 1
#   cp <- mallows.cp(obs_vec, design_mat, idx_vec, invgv_mlr_var)

#   repeat{
#       # vector of selection
#       sel_vec = setdiff(full_idx_vec, idx_vec)

#       if (length(sel_vec) == 0){
#           break
#       }

#       tmp_idx_vec_1 <- c(idx_vec, sel_vec[1])
#       tmp_cp_1 <- mallows.cp(obs_vec, design_mat, tmp_idx_vec_1, invgv_mlr_var)

#       for (i in 2:length(sel_vec)){
#           tmp_idx_vec_2 <- c(idx_vec, sel_vec[i])
#           tmp_cp_2 <- mallows.cp(obs_vec, design_mat, tmp_idx_vec_2, invgv_mlr_var)

#           if (tmp_cp_2 <= tmp_cp_1){
#               tmp_cp_1 <- tmp_cp_2
#               tmp_idx_vec_1 <- tmp_idx_vec_2
#           }
#       }

#       if (cp >= tmp_cp_1){
#           cp <- tmp_cp_1
#           idx_vec <- tmp_idx_vec_1
#       }else{
#           break
#       }

#       # debug information
#       print(idx_vec)
#       print(cp)
#   }

#   # return
#   idx_vec
# }

# model selection: simulated annealing: neighbour function
ms.sa.nbr = function(idx_vec){
    # get random index (1 is not used)
    rand_idx <- sample(2:gv_par_size, size = 1)

    if (rand_idx %in% idx_vec){
```

```
        # delete rand_idx in idx_vec
        nbr_idx_vec <- idx_vec[idx_vec != rand_idx]
    }else{
        # add rand_idx to idx_vec
        nbr_idx_vec <- c(idx_vec, rand_idx)
    }

    # return
    nbr_idx_vec
}

# model selection: simulated annealing: probability function
# costs will be minimized
ms.sa.prob <- function(old_cost, new_cost, temp){
    # return
    exp( (old_cost - new_cost) / temp )
}

# model selection: simulated annealing
ms.sa = function(idx_vec = c(1), temp = 100, alpha = 0.99, it_max = 10000, it_exit = 1200){
    old_cost <- ms.cp(idx_vec);
    it_same <- 0

    for (i in 1:it_max){
        nbr_idx_vec <- ms.sa.nbr(idx_vec);
        new_cost <- ms.cp(nbr_idx_vec)

        if ( ms.sa.prob(old_cost, new_cost, temp) >= runif(1) ){
            idx_vec <- nbr_idx_vec
            old_cost <- new_cost
            it_same <- 0
        }else{
            it_same <- it_same + 1
            if (it_same >= it_exit){
                break
            }
        }

        temp <- alpha * temp

        # debug
        # print(idx_vec)
        # print(old_cost)
        # print(temp)
    }

    # return
    idx_vec
}

#
ms.sim = function(expect_vec, var, sim_count = 10){
    sd <- sqrt(var)
    spse <- 0

    for (i in 1:sim_count){
        # generate and init pseudo observables
        gv_obs_vec <<- rnorm(gv_sample_size, expect_vec, sd)
        gv_transf_obs_vec <<- t(gv_design_mat) %*% gv_obs_vec
        mlr.init()

        # select model
        idx_vec <- ms.sa()
        tmp_spse <- ms.rss(idx_vec) + (2 * gv_mlr_var * length(idx_vec))

        # calculate spse
```

```
        spse <- spse + tmp_spse

        # debug
        print("sorted index vector:")
        print(sort(idx_vec))
        print("tmp spse:")
        print(tmp_spse)
        print("tmp mallows' cp:")
        print(ms.cp(idx_vec))
    }

    spse <- spse / sim_count

    # return
    spse
}
```