Nonlinear Multiple Regression Methods for Spectroscopic Analysis:Application to NIR Calibration

Chenhao Cui

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Statistical Science

University College London

October 8, 2018

I, Chenhao Cui, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

Chemometrics has been applied to analyse near-infrared (NIR) spectra for decades. Linear regression methods such as partial least squares (PLS) regression and principal component regression (PCR) are simple and widely used solutions for spectroscopic calibration. My dissertation connects spectroscopic calibration with nonlinear machine learning techniques. It explores the feasibility of applying nonlinear methods for NIR calibration.

Investigated nonlinear regression methods include least squares support vector machine (LS-SVM), Gaussian process regression (GPR), Bayesian hierarchical mixture of linear regressions (HMLR) and convolutional neural networks (CNN). Our study focuses on the discussion of various design choices, interpretation of nonlinear models and providing novel recommendations and insights for the construction nonlinear regression models for NIR data.

Performances of investigated nonlinear methods were benchmarked against traditional methods on multiple real-world NIR datasets. The datasets have different sizes (varying from 400 samples to 7000 samples) and are from various sources. Hypothesis tests on separate, independent test sets indicated that nonlinear methods give significant improvements in most practical NIR calibrations.

Acknowledgements

First of all, I would like to express my gratitude to Prof. Tom Fearn for guidance and advice on my research. As my mentor, he supports me at every stage of this project with thoughtful inputs and through many inspiring interactions. His knowledge, experience and patience have been critical to the success of this project. It was also my fortunate to have had the chance to work with him. I would also like to thank many excellent staff and Ph.D student colleagues in the statistical science department in UCL, for graciously contributing time and efforts on my project.

I would like to acknowledge the sponsorship of the Ph.D project from Buhler AG, who provides financial support and invaluable NIR datasets for this study. In particular, I would like to thank Martin Heine and Juste Hahne for demonstration of the laboratory and industrial systems, and for the fruitful discussions on the applications. I would like to thank my colleagues in Buhler UK, especially Hamid Gabriel, Timothy Kelf and Jean-Francois Deprez for stimulating discussions. It was a great experience to work with the London team. In addition I want to express my gratitude to my colleagues in EPFL, for assistance during my stay in Lausanne.

My sincere thanks goes to Matthias Graeber, who coordinates my project with the company, provides me with invaluable chances to learn and grow. His reviews, domain knowledge and endless patience have been critical for this project. Matthias has been a excellent manager, and also a good friend of mine for the past four years.

I would like to warmly thank my parents, for unquestioning support and belief

in me. My study truly would not have been possible without your help and encouragement.

Finally big thanks to many people I have met during my Ph.D study who helped me along with my academic journey and my life, without your gracious support, it would not have been such an enjoyable experience. I dedicate this work to you! Wish all of us a tremendous new adventure.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this thesis, we introduce three different nonlinear regression schemes for multivariate spectroscopic analysis: kernel methods including least squares support vector machines (LS-SVM) and Gaussian process regression (GPR), hierarchical mixture of linear regressions (HMLR) and convolutional neural networks (CNN). These methods are introduced to improve the performances of predictive models on NIR datasets. In the following content, our main discussions are focused on: 1) compare the predictive performances of our introduced nonlinear regression methods with the benchmark method of PLS regression on real-world NIR datasets. 2) provide insights and recommendations on construction of these nonlinear models. 3) discuss the relationships and differences between three nonlinear regression methods, explore the feasibility of using our proposed methods in real-world applications.

In Chapter two we review principles, history and applications of near infrared (NIR) spectroscopy. Benchmark regression methods of PLSR and PCR are introduced. Relevant statistics and mathematics frequently used in the whole thesis, including hypothesis test, evaluation of regression coefficients and Fourier transform are also presented in detail.

In Chapter three we describe in detail all of the datasets used in our studies. We also introduce software and programming environments that are involved in

developing calibration algorithms.

In Chapter four we present two kernel regression methods — LS-SVM and GPR. The performance of the two proposed non-linear regression methods is assessed and compared to the benchmark regression method of PLS regression on a NIR dataset of whole wheat grains. LS-SVM and GPR both show enhanced generalization performance, especially for maintaining homogeneous predictive precision over the range. LS-SVM and GPR have very similar regression strategies and predictive performance on our dataset. We also address the issue of choosing the loss function and optimization method in kernel methods. Results indicate that when the training set is small, leave-one-out cross-validation (LOO-CV) combined with a squared error loss function outperforms gradient based evidence maximization (EM) for GPR. In this study, our main contributions to knowledge include 1) careful comparisons between LS-SVM, GPR and PLSR on a real NIR dataset; 2) detailed discussion of impacts of various parameter tuning methods (and inference methods) in kernel regression algorithms.

In Chapter five we investigate the use of the HMLR and variational inference (VI) for multivariate spectroscopic calibration. The performance of the HMLR method is compared to the PLS regression, and PLS based locally weighted regression (LWR) on three different NIR datasets. In these comparisons, HMLR outperforms the other two benchmark methods in predictive performance. Compared to LWR, HMLR is parametric, which makes it interpretable and easy to use. In addition, we demonstrate that HMLR can automatically split the dataset into two subsets based on constituent concentration in a probabilistic way and build independent PLSR models on each of them. This is found especially useful when the investigated constituent covers a large range. In summary, we propose a novel structure for linear model ensemble in this study. We introduce how to use VI to train the model, and how to interpret and make predictions with the model. The performance of the method is validated on 3 industrial NIR datasets.

In Chapter six we explore the use of convolutional neural networks (CNN) for NIR calibration. We propose a unified CNN structure that can be used for general multivariate regression purpose. The comparison between the CNN method and the PLS regression method is done on three different NIR datasets. We demonstrate that the CNN models are more precise and less noisy than the PLS regression models. The convolutional layer in the CNN model can automatically find the suitable spectral preprocessing filter on the dataset, which saves a lot of time when developing new calibrations. The final regression equations and the intermediate transformations of the CNN models are plotted for a visual comparison with the PLS regression models. In this study our main contributions to knowledge include: 1) we propose the method of using CNN on NIR regression tasks, which can automatically generate suitable pretreatment function on investigated dataset; 2) we introduce a way to visualize the ”black-box” CNN models, for the purpose of interpretation; 3) we carefully discuss the influences of different design choices, model hyperparameters, and provide recommendations and insights for construction CNN models on NIR datasets.

In the final chapter we summarize the results obtained and our main contribution to the knowledge of nonlinear spectroscopic regression methods. We discuss the strengths and weaknesses of the introduced nonlinear regression methods. Finally we also provide recommendations for the users to choose a suitable regression scheme for practical applications.

Chapter 2

Reviews of near-infrared spectroscopy, multivariate regression and other relevant statistical methods

2.1 Principles, history and applications of near infrared spectroscopy

2.1.1 Spectral interpretation of NIR bands

In general, spectral vibrations in NIR region do not correspond with any known fundamental vibrations. Overtones, combination bands and Fermi resonances together contribute to the difficulty on interpretation of NIR bands. In what follows we will briefly introduce the origins of these bands.

First we should appreciate how the spectral bands are linked to the potential energy of vibration modes. The easiest way to describe the vibration mode of an oscillator is to treat it as a harmonic oscillator. Figure 2.1 shows the relationship between the potential energy of an oscillator and its inter-nuclear separation, and the red dotted line describes a harmonic oscillator. If we denote ω_i as the vibrational frequency of the i^{th} mode, v_i as the vibrational quantum number, we can express the potential energy for a harmonic oscillator as:

$$E_i = \omega_i(v_i + \frac{1}{2}). \tag{2.1}$$

Fundamental vibration occurs when the oscillator transit from v_0 to v_1. Overtones are not allowed under the harmonic assumption.

However, a more realistic assumption is to treat the molecule as an anharmonic oscillator, refer to the black solid line in Figure 2.1. Potential energy of an anharmonic oscillator can be described by:

$$E_i = \omega_i x_i(v_i + \frac{1}{2})^2, \tag{2.2}$$

where x_i is the anharmonicity constant. As we can see from Figure 2.1, the correction on the potential energy of an anharmonic oscillator becomes larger for higher excited states. Under the anharmonic assumption, the fundamental transition from v_0 to v_1 is still dominating, but the transition on $v_0 \rightarrow v_2, v_0 \rightarrow v_3, \ldots$, which are the overtones in spectral bands, can occur. Overtones have lower frequencies than multiples of the fundamental band, for example $w_{v_0 \rightarrow v_2} < 2 \times w_{v_0 \rightarrow v_1}$. Overtone bands have lower intensities than the fundamental band in general.

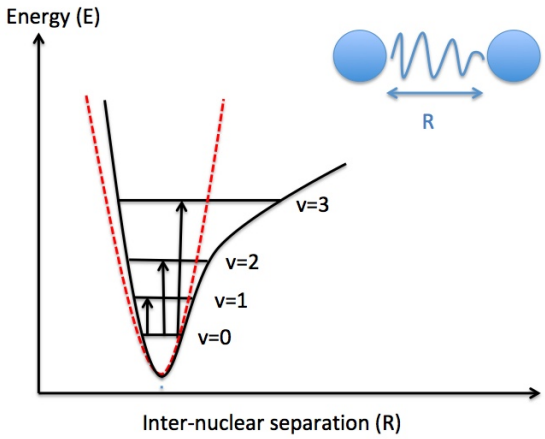


Figure 2.1: Potential energy (E) vs. inter-nuclear separation (R) for a harmonic oscillator (red dotted line) and an anharmonic oscillator (black line).

Combination bands are also commonly observed in NIR spectrum. Combi-

nation bands occur when two fundamental vibrations are excited simultaneously. Combination bands occur under two strict conditions: the two fundamental vibrations must come from the same functional group; the two fundamental vibrations must have the same symmetry. A typical example is the combination band on CH_2. Figure 2.2 shows the two fundamental vibrations, i.e. symmetric stretching and scissoring, that can couple to form a combination band.

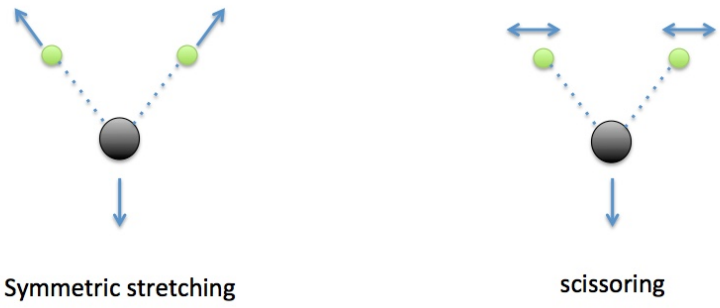


Figure 2.2: Combination band of CH_2: symmetric stretching and scissoring.

Another factor that makes the NIR spectrum even more difficult to interpret is Fermi resonance. Fermi resonance occurs when two vibration bands that have the same symmetry also have similar intensities and frequency. The effect of Fermi resonance can be illustrated by Figure 2.3. The intensity of the relatively weaker band becomes higher and that of the stronger band becomes lower. As a result, Fermi resonance equalizes the intensity of the two bands. In addition, the frequency gap between the two bands also becomes larger, which means the spectral positions where the two bands originally occur are shifted. Fermi resonance is frequently used to explain unknown bands in the NIR spectrum, unfortunately it can be easily mis-assigned.

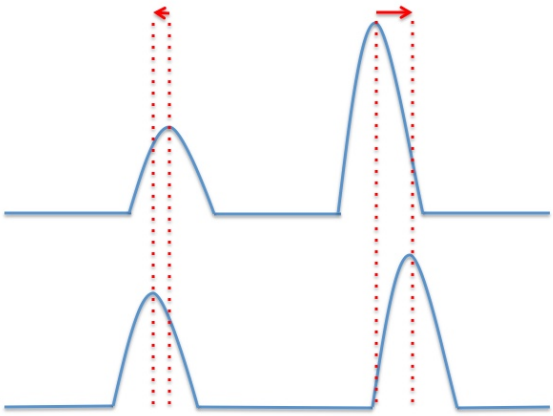


Figure 2.3: Fermi resonance that causes intensity borrowing and frequency shift.

In summary we introduced three factors that can complicate the interpretation of near infrared bands: overtones, combination bands and Fermi resonance. It is worth noting as chemometricians we should not over-interpret the spectrum or the model we get. Understanding the principles of NIR spectrum helps us correlate the result we get with the physical and chemical theory, however, it should only be used as a reference instead of a criterion to evaluate our results.

2.1.2 Interaction of NIR radiation with particles in a sample

At the very early stage of NIR applications, the technique was proposed as a rapid method for prediction of moisture content in wheat flour[1]. We should notice there are many different types of interaction between NIR radiation and particles in measured samples, including reflectance, transmittance, refraction and scattering, etc. See Figure 2.4 for an example. At the very early stage of applications, transmittance was used in NIR calibration, with samples dissolved in carbon tetrachloride. NIR measurements was collected in the form of $\log(1/T)$ due to Beer's law, which relates absorbance to concentration for dilute solutions. Later the researchers realized that the design of experiment was not appropriate, and then they started using diffuse reflectance for NIR spectrum acquisition. The same group of engineers used the same design with logarithmic photometers, which led to NIR measurement on $\log(1/R)$. The users found that $\log(1/R)$ achieves good predictive accuracy, as a result it has become the most popular way to record NIR measurement.

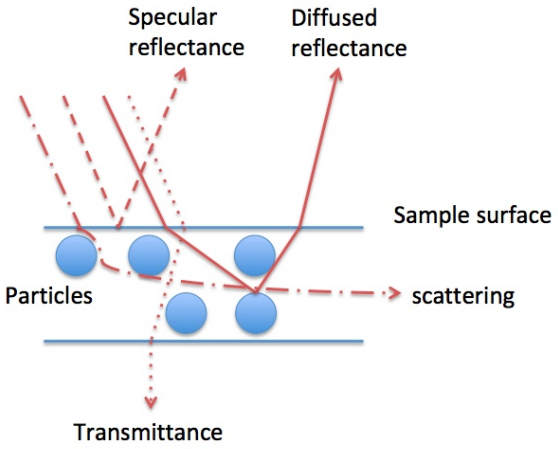


Figure 2.4: Different types of interaction of NIR radiation with particles in a sample

2.1.3 A review of chemometrics methods for NIR calibration

The initial form of chemometrics was dual-wavelength calibration. For example, Birth (1960)[2] reported the detection of discoloured potato tissue by transmitted light. They observed that the optical density difference (ΔOD) of two selected wavelengths (800 *nm* and 710 *nm*) is an indicator of discoloured potato tissue. The dual-wavelength calibration technique was further discussed by Cowles (1965)[3]. However, they found that for turbid samples it is easier to interpret the experimental data by using the ratio instead of the linear difference. In the same paper, they also mentioned the ratio detection method is also useful for correcting small absorption changes under large background scattering effects.

At the same time, calibrating spectroscopic data for the chemical composition in near infrared region was initialised by Karl Norris. The instrument available at that time, namely Cary 14 (and Cary 14B), was designed for UV and visible regions, with the NIR measurement as an additional functionality. Such kinds of spectrophotometers measure an essentially continuous spectrum in the NIR region, with a relatively low signal to noise ratio (SNR) by present standards. However, the original form of NIR calibration was still dual-wavelength calibration. Norris and Hart (1965)[4] published their initial work on determination of moisture content of

agricultural products. They measured light transmittance of wheat (flour, bran and whole grain) and soybean, found that ΔOD at 1960 *nm* and 2080 *nm* is correlated to moisture content. However, the optimum wavelengths for prediction of moisture depend on the measured products. For example, for peanuts, the two wavelengths that are the best for prediction of moisture content are 970*nm* and 900*nm* as reported. Continuing work from Ben-Gera and Norris (1968)[5] extended the field of research to other food products, with the same calibration method, namely dual-wavelength calibration. In the paper they interpret data w.r.t absorption from O-H and C-H stretching vibrations and found that for meat samples, ΔOD of 1725 *nm* and 1800 *nm* can be used to determine moisture, while fat content is correlated with ΔOD of 1650 *nm* and 1725 *nm*.

There were very few NIR applications at that time. The main reasons are: 1) The Cary 14 had a poor scanning speed. It could take up to 30 minutes to acquire one measurement; 2) searching for useful calibration wavelengths requires lots of knowledge and manual work; 3) working in the transmittance mode significantly limited the range of products (it is challenging to acquire the adequate signal for thick, less transparent products). Some samples need special preparation before measurement, which is inconvenient. The 1970s saw the rapid growth of NIR industry. Reflectance spectrophotometer had become popular in a short space of time. Initial work on construction such kind of systems was from Massie and Norris (1965)[6]; Besides, manufactures including Dickey-John, Neotec and Technicon started producing commercial spectrophotometers specially designed for industrial NIR applications. Industrial instruments at that time mostly used interference filters instead of a monochromator to generate NIR wavelengths. For example in 1971, Dickey-John produced the first generation of grain analysis computer (GAC), which uses six filters ranged from 1680 *nm* to 2310 *nm* to generate NIR wavelengths. Following generations of GAC extended the number of filters to 10 (Dickey-John GAC III, 1977). Technicon produced an extended system in a similar way, namely Infra-Alyzer 400R, which employed 19 filters. Scanning NIR spectrophotometers were

gradually developed at the same time. Neotec started with a replica of the Cary 14 in 1971, then continuously improved the design and the performance of the system. Landa and Norris (1979)[7] published their work on an improved design of NIR measurement system based on interferometer spectrophotometer. The introduced system was able to rapidly measure parameters including stray light, resolution, accuracy, etc., of wavelength readout. Neotec soon commercialized the system as model 6350. Three years later (1982) Neotec updated the system with the Northstar computer (model 6250). In the meanwhile, Technicon also produced a scanning spectrophotometer — InfraAlyzer 500 in 1979. Such kind of scanning spectrophotometers (sometimes termed spectrocomputer when equipped with computer), can obtain readouts on the whole spectrum instead of a few wavelengths generated by fixed interference filters. Researchers at that time reported benefits of such flexibilities in some applications. For example, Connell and Norris (1980)[8] claimed that when measuring washed wool, a small improvement was obtained with non-standard wavelengths.

Chemometrics were improved simultaneously. Stepwise multiple linear regression (MLR) replaced the traditional treatment (optical density difference) in NIR calibrations. Pioneer work from Norris, Barnes, Moore and Shenk(1976)[9] established the application of the method on prediction of forage quality. In the paper they exhibited a way of analysing high dimensional data. In the research NIR measurement ranged from 1400 *nm* to 2400 *nm* with a resolution of 0.5 *nm*. They first smoothed the spectra by averaging adjacent points to reduce the number of explanatory variables to 500, then applied stepwise MLR to choose 9 variables. In the research they also reported that using the second derivative of $\log(\frac{1}{R})$ gives much better prediction results. The idea was also discussed by Osborne, Douglas, Fearn (1983)[10]. In the paper of Osborne et al, they explained that *log* values have low correlation with anything other than the particle size of the sample. As a result, in a stepwise procedure, selection of the first variable must be almost random. The method soon spread to other applications. Law and Tkachuk (1977)[11] presented

their successful results of using the diffuse reflectance spectrum of wheat grains to predict their compositions (protein, moisture, etc.). McClure, Norris (1977)[12] used diffuse reflectance of tobacco samples to predict their sugar content. To improve the method, Osborne, Fearn, Miller and Douglas(1984)[13] presented other wavelength selection methods, based on picking wavelengths in pairs across the whole spectrum and backward stepwise wavelength selection from a chosen subset of variables.

The main limitations of MLR on spectroscopic calibration are multicollinearity and near-multicollinearity. The issue was discussed in detail by Martens and Næs (1989)[14]. When there are more explanatory variables than samples, which is the typical case for a scanning spectrophotometer, the matrix $\mathbf{X}^T\mathbf{X}$ in the ordinary MLR equation is singular. As a result, the least squares solution is not unique anymore. Even when the number of samples increases, explanatory variables in $\mathbf{X}$ are still mutually dependent. Multicollinearity and near-multicollinearity often lead to unsatisfactory or incorrect solutions.

One solution to the issue is ridge regression. The method was first proposed by Hoerl and Kennard (1970)[15]. They addressed the issue that the least squares treatment could be inappropriate for some regression problems. They claimed that the estimator could be improved by adding the ridge trace term to $\mathbf{X}^T\mathbf{X}$. To follow the discussion, Goldstein and Smith (1974)[16] examined the mean squared errors (MSE) property of ridge regression. They reported that the ridge estimator, which is biased, has smaller MSE in comparison with the least squares solution. However, Fearn (1983)[17] argued that when working on the spectroscopic dataset, ridge regression does not work in the obvious way. The main spoiler is that the major variance in *log* values of reflectance spectrum is highly correlated with the particle size, which is irrelevant with the main regression target. Hoerl, Kennard and Hoerl (1985)[18] responded that by removing the artificial correlations in the reflectance spectrum, the resulting system becomes near-orthogonal. They claimed that when

working on transformed data, ridge regression could deliver better prediction results in randomised tests. Geladi, Macdougall and Martens (1985)[19] discussed the same issue. In the paper they suggested using another spectrum transformation approach: multiplicative scatter correction (MSC), which separates chemical light absorption from physical light scatter in NIR spectrum.

The other treatment for multicollinearity is data compression, where the number of explanatory variables is reduced before MLR. Wavelength selection is a typical one-stage data compression method. However, considering the many inconveniences and limitations as discussed above, the more popular data compression methods in literature are two-stage methods, namely principal components regression (PCR) and partial least squares (PLS) regression. Both of these regression methods first shrink the number of input variables, and then perform a MLR.

Cowe and McNicol (1985)[20] published their initial work on the use of PCR for NIR calibration. They demonstrated the regression method on calibrating moisture and protein concentration of wheat flour and water extract values on milled barley. They addressed the issue of intercorrelation for NIR calibration, and presented the most appealing point of PCR in the context of spectroscopic analysis: interpretation of data. They showed that PC1 could be interpreted as the particle size of measured samples, PC2 of wheat and PC3 of barley spectra have a very similar shape to the spectrum of water, PC4 has the same peak as that used in filter instruments for measuring protein in cereal. Næs and Martens (1988)[21] extended the discussion on PCR from a theoretical viewpoint. They carefully viewed PCR in comparison with LS regression, pointed out the advantage of PCR on small datasets. They decomposed prediction error to a bias term and a variance term (see also Næs, Irgens and Martens (1986)[22]). They explained that LS method can have no bias, i.e. no underfitting error, which is practically inappropriate. PCR delivers smaller MSE by allowing for bias in regression. In the same paper, they also provided insights on why deletion of small eigenvalue components is sensible.

They interpreted small eigenvalue terms as directions associated with noise from the instruments and the data acquisition process. In addition, small eigenvalues are less stable and representative of population variation. They also illustrated how components could be selected by a novel cross-validation method introduced by Martens and Næs (1987)[23] and explained how PCA can function as an outlier detection method.

PLS regression was proposed at the same period. Wold (1975)[24] pioneered the research on NIPALS algorithm. Wold, Martens and Wold (1983)[25] for the first time used PLS regression on chemical applications. In the paper, they explained the PLS algorithm as a similar data compression approach to PCR, with the ability to detect structures in $\mathbf{X}$ with the relevant predictive covariance with $\mathbf{y}$. They also mentioned an additional feature of PLS: checking new samples according to the calibration set, which could be viewed as a solution for outlier detection. Martens and Jensen (1983)[26] for the first time applied PLS regression on an NIR dataset. Wold, Ruhe, Wold and Dunn (1984)[27] continued the discussion on the PLS method. They compared PLS regression with RR and PCR on chemical examples, highlighting the collinearity issue. They also presented the appropriate way to select components by cross-validation. Næs, Irgens and Martens (1986)[22] gave a nice comprehensive comparison of different linear regression methods for NIR calibration. They addressed the issue when the number of explanatory variables are more than the number of samples, biased estimators including PCR and RR down weight information associated with specific eigenvectors, hence improving robustness and stability of the model. PLS regression, compared to PCR, uses fewer components and get the same performance. They argued it is essential to use fewer factors from the standpoint of model interpretation. They encouraged using PLS for a wider range of practical applications.

PCR and PLS regression soon became very popular. At present they are still the dominating "standard" solution for multivariate calibrations. It is worth not-

ing that another type of data compression method, Fourier transformation, was proposed at the same period. McClure and Giesbrecht (1984)[28] proposed using Fourier transform on NIR calibration. They carried out experiments on tobacco samples. 11 coefficients in the Fourier domain were fed into a stepwise MLR. They claimed the method is much faster in selecting variables and more compact than the ordinary stepwise MLR. By removing the mean of Fourier terms, effect of particle size can be corrected. They claimed Fourier transform could be used for data reduction, spectral smoothing, calibration transfer, etc. Davies and McClure (1985)[29] continued study on the topic with the target of classification of instant coffee samples. By using Fourier transform they better discriminated normal and decaffeinated instant coffee samples, without sample preparation. A more interesting feature of Fourier transform was reported by McClure and Davies (1988)[30]. In the paper, they showed an example of using Fourier self-deconvolution for signal enhancement. However, compared to PCR and PLS regression, Fourier transform provides no interpretation or visualisation in the wavelength domain, which makes it less attractive in practical applications.

Nonlinear regression methods on NIR calibration can be traced back to the 1980s. A lot of exciting discussions arose on techniques such as support vector machines (SVMs), locally weighted regression (LWR), artificial neural networks (ANNs), etc. We will review previous works on these topics in the following chapters in a greater detail.

2.1.4 Applications of NIR spectroscopy

2.1.4.1 Cereals

In the early stage of NIR development, most of the successful stories came from the agricultural industry. Norris and Hart (1965)[4] first demonstrated the possibility of using NIR spectroscopy to predict moisture content of various grain samples. From the 1970s commercial instruments including Dickey-John GAC, Neotec tilting-filter spectrometer and Technicon InfraAlyzer had been developed for NIR measurement in reflectance mode. These instruments were built with either fixed filters or scan-

ning monochromator. Williams (1975)[31] reported calibrations on moisture, crude protein (CP) and oil content in cereal grains and oilseeds by using NIR reflectance. He reported that the range of standard errors on moisture prediction was from 0.12% (barley) to 0.30% (rapeseed); standard error on predicting CP was reduced from 0.30% to 0.15% when mixing samples 15 times. Stermer (1977)[32] used NIR reflectance from two tilting filter instruments to predict the moisture content of maize and sorghum grain. The standard errors were 0.8% and 3.4% for maize and sorghum respectively. R Tkachuk (1981)[33] introduced a study of the prediction of oil and protein content in rapeseed, based on the reflectance instrument the Carry 17. In 1980, the first commercial NIR instrument solely designed for whole grain, Trebor GT-90, was produced. Different from other instruments, Trebor GT-90 was developed to work in the transmittance mode. It employed 12 filters in a wavelength range of 850-1050 nm, which is shorter than the common wavelengths selected for reflectance instruments. Williams, Norris and Sobering (1985)[34] carefully discussed the performance of Trebor GT-90. Williams and Sobering (1993)[35] performed a comparison on instruments based on reflectance mode and transmittance mode, with an extended discussion on wavelength optimization when calibrating on different constituents of whole grain samples. Osborne et al. (1982)[36] carried out a feasibility study on building universal calibration for protein and moisture of wheat with different varieties and harvested from different years. Their test, based on Technicon InfraAlyzer 2.5, showed that it is possible to build a universal calibration for different types of wheat across 5 years, except for one year (1976) when abnormal climate prevailed. Manley et al. (2002)[37] presented results on prediction of hardness, protein and moisture content in whole wheat grains with a FT-NIR system. PLS regression models were built on MSC and baseline corrected spectra. They reported the RMSEPs for hardness, protein and moisture predictions were 2.13, 0.51 and 0.15% respectively. Pettersson et al. (2003)[38] explored the feasibility to use NIR transmittance for determination of mycotoxins in cereal. By applying PLS regression and PCR on a spectral range of 670 -1100 nm, they obtained a good correlation coefficient (0.984) and satisfactory standard error ($381\mu g$

DON per Kg). However, relatively large numbers of factors (11-13) in the PLS and the PCR models were required to build such kind of calibrations. They concluded it might be possible to identify kernels with DON concentration above European limits for flour.

There are also many applications of NIR spectroscopy on flour samples. Watson et al. (1976)[39] evaluated the impacts of different NIR instruments (fixed-filter or scanning monochromator) and different grinders on prediction of protein content of wheat. They reported that the influence of grinder was more significant than NIR instruments. Osborne, Douglas and Fearn (1982)[40] reported using NIR reflectance to predict protein, moisture, particle size, colour and starch damage of flour. Hareland (1994)[41] further investigated the application of NIR reflectance for particle size measurement on different types of wheat, and compared the results from NIR to laser diffraction and sieve analysis. Osborne and Fearn (1983)[42] demonstrated for protein and moisture calibration transfer between different instruments of the same model. They circulated the same 20 samples across different laboratories to adjust the calibration constant, then circulated another 20 samples to validate the model. They reported on the second set of samples, accuracy on protein prediction was 0.13-0.33% with a precision of 0.07%; the accuracy for moisture prediction was 0.20-0.29% with a precision of 0.05%. Diachuk et al. (1981)[43] and Iwamoto et al. (1984)[44] reported on prediction of ash content of flour with NIR reflectance analysis. According to Iwamoto's results, standard errors of prediction of ash were 0.035% and 0.031% for two different flour samples. However, it was believed that ash has a secondary correlation with NIR spectrum. Ash itself does not have any specific NIR characteristic, but is correlated to oil, protein and particle size. Consequently Ash calibrations share wavelengths with other properties. Miralbes (2004)[45] demonstrated the potential of NIR reflectance on determination of a wider range of constituent concentration, including protein, moisture, dry gluten, wet gluten, starch damage and ash contents.

In addition to determination of constituent composition and physical characteristics, NIR spectroscopy can be used to detect cereal damage and infection of cereals. For example, Dowell et al. (1998)[46] developed an automated single grain analysis system to detect internal insect damage on wheat grains. They claimed the introduced system was not sensitive to the moisture, protein content or varieties of wheat. A detection accuracy of 95% could be achieved on rice weevil infection. Perez-Mendoza et al. (2003)[47] reported on detection of insect fragments in wheat flour by NIR spectroscopy. They concluded that NIR calibration could be developed to predict whether the insect fragments in flour samples exceeds 130, but was unable to predict the insect fragments at the FDA action level. Singh et al. (2009 and 2010)[48][49] used a hyper-spectral imaging system to detect insect-damaged whole wheat grains. They reported that by using a quadratic discriminant analysis (QDA) classifier, discriminate accuracy of 96.4% on healthy grains and 91.0%-100% on insect-damaged grains could be achieved. Singh et al. (2012) further reported on classification on fungal damaged wheat grains with a linear discriminant analysis (LDA) model on hyper-spectral NIR images. They reported an accuracy of 97.3-100% on identification of fungal-infected wheat grains.

2.1.4.2 Forage

Forage and feedstuff samples are usually mixtures of multiple components. Each of the components has its specific NIR absorption characteristic, which makes NIR spectroscopy useful for forage analysis. Norris and Barnes (1976)[9] first reported on using NIR reflectance from a scanning NIR instrument (Cary model 14) to analyze the quality of forage. They measured NIR reflectance from alfalfa, tall fescue and alfalfa bromegrass mixtures, then used MLR to select 9 wavelengths for the prediction of CP, neutral detergent fiber (NDF), acid detergent fiber (ADF), lignin (L),in vitro dry matter disappearance (IVDMD), in vivo digestibility (DMD), dry matter intake (DMI) and digestible energy intake (DEI). According to their results, the standard errors of estimate were 0.95% for CP, 3.1% for NDF, 5.1% for DMD and 7.9 g for DMI. Barton and Burdick (1983)[50] also reported on using scanning NIR instruments for the prediction of DMI in bermudagrass hays, with a standard

error on calibration of 1.78% and a standard error of analysis of 2.54%.

There were also many researches on fixed filters NIR reflectance instruments. Shenk and Barnes (1977)[51] claimed that using the whole spectrum, rather than a few fixed filtered wavelengths in existing instruments, smaller prediction errors on CP and IVDMD could be obtained. Counts and Radloff (1979)[52] reported similar standard errors of analysis on IVDMD from a six-filter NIR instruments, compared to the laboratory scanning instruments. Winch and Major (1981)[53] carried out researches on the prediction of Nitrogen (N), IVDMD and DMD for grasses, legumes and legume grass mixtures with a six-filter instrument (Technicon InfraAlyzer 2.5). They reported low standard errors of calibration when analyzing N, but relatively large errors for IVDMD and DMD. Multiple researches indicated that various considerations must be addressed to obtain satisfactory results. For example, Barton and Burdick (1979)[54] reported calibration results on CP, NDF, ADF, L, IVDMD with a tilting-filter NIR reflectance instrument. They addressed that warm and cool-season grasses should be analyzed separately, with different regression functions. Fales and Cummins (1982)[55] reported that high humidity in Sorghum forage samples could introduce extra error on estimating ADF content. Minson et al. (1983)[56] reported their results based on a 19-filter Technicon NIR reflectance instrument. They indicated that to remove the biases on prediction of CP, IVDMD and voluntary intake (VI), grass species, plant part and physical form of the tested sample must be taken into consideration.

2.1.4.3 Dairy

Dairy products were studied at the very early stage in NIR history. Ben-gera and Norris (1968)[57] presented their results on calibrating fat content of milk on NIR reflectance. Casado et al. (1978)[58] extended the study to fat, protein and moisture in dried milk. Vilder and Bossuyt (1983)[59] carried out a similar research on prediction of protein, fat and moisture content in milk powder, using a Technicon InfraAlyzer 400, a NIR reflectance instrument with 19 filters. Baer et al. (1983)[60] also reported their results on Technicon InfraAlyzer 400, where they

picked 3 wavelengths for the prediction of moisture, 4 wavelengths for fat, 4 wavelengths for lactose, 8 wavelengths for micro-Kjedahl protein and 7 wavelengths for dye binding protein. The standard error of prediction (SEP) values are 0.274%, 0.099%, 0.594%, 0.438% and 0.509% respectively. Frankhuizen and van der Veen (1985)[61] reported on prediction of ash and lactate contents in milk powder, with the same type of instrument. NIR reflectance could also be used for classification of milk powder, according to Downey et al. (1990)[62], where they classified milk powder by the thermal treatment received. PCA and PLS regression had impacted NIR analysis on diary products since proposed. Robert et al. (1987)[63] applied PCA and factorial analysis on NIR reflectance spectrum of milk and found 1724 *nm*, 1752 *nm*, 2308 *nm* and 2344 *nm* are highly correlated with fat content, 2050 *nm* and 2180 *nm* are indicators of protein, and the large absorption band at 2094 *nm* could be used for discrimination of samples by lactose content. Laporte and Paquin (1999)[64] reported satisfactory results when PLS regression was used on prediction of fat and nitrogenous constituents in milk. SEP values were 0.07%, 0.06% and 0.05% correspondingly for fat, crude protein and casein, true protein. They also notified that homogenization had a positive impact on prediction accuracy.

NIR has been reported as a powerful tool when analyzing butter and cheese. Giangiacomo et al. (1979)[65] reported high correlations between NIR reflectance and protein, ash, free tyrosine, PH value and fat in blue cheese, based on their research with a fix-filter NIR instrument. Frank and Birth (1982)[66] investigated 30 different cheese samples, including Parmesan, Gruyere, Romano, Cheddar, and Colby. They found the spectral range of 1170 *nm* to 1350 *nm* and 1640 *nm* to 1710 *nm* are highly relevant for the prediction of fat, protein and moisture. Pillonel(2003)[67] demonstrated the classification of cheese from six different origins by applying principal component linear discriminant analysis (PC-LDA). Sørensen and Jepsen (1998)[68] reported the results of calibrating NIR spectrum on sensory properties of cheese. They compared reflectance spectroscopy (with a spectral range of 1100 *nm* - 2490 *nm*) with transmittance spectroscopy (850 *nm*

- 1050 *nm*). They found that accuracies for prediction of consistency and flavour properties with reflectance spectroscopy are generally better than transmittance spectroscopy. They also indicated that the squared correlation coefficients (R^2) on consistency attributes, obtained by PLS regression on NIR reflectance, are significantly higher than that of flavor attributes. González-Martín (2007)[69] employed a remote reflectance fibre-optic probe for NIR acquisition. The introduced system was used to predict the percentage of cow's, ewe's and goat's in cheese with different ripening time. SEC values of 11.6%, 10.6% and 9.8% were obtained respectively.

NIR calibration can be used on whey products. Baer et al. (1983)[60] applied NIR reflectance on compositional analysis of different types of whey powders, and concluded that the samples had similar constituent concentration. Pouliot et al. (1997)[70] assessed the potential of a remote optic fiber probe integrated NIR spectrophotometer system on prediction of whey changes during processing. PLS calibrations were built for evaluation of heat denaturation of proteins, chemical composition of whey protein concentrate (WPC) during production, and the degree of hydrolysis (DH) of whey proteins during trypsin hydrolysis. They varied the production and the processing conditions to ensure the reliability of the calibration models. Nørgaard et al. (2005)[71] reported satisfactory results on prediction of crystallinity of lactose in whey permeate powder with NIR calibration. They selected five wavelength intervals in the spectral range of 1100 *nm* - 2498 *nm* then performed PLS regression on the selected wavelengths. RMSECV was 0.27% on 35 tested samples as reported. The accuracy of the NIR calibration was also better than a Raman calibration obtained from the same sample set.

2.1.4.4 Meat

One of the biggest markets of NIR spectroscopy is on meat and meat products. Prices of meat/meat products vary significantly according to homogeneity and quality characteristic. Ben-gera and Norris (1968)[5] investigated fat and moisture content in 2 *mm* thick meat samples by NIR transmittance spectrum. The NIR

absorption were interpreted as O-H and C-H stretching vibrations together with light scattering losses. Dual-wavelength calibrations were constructed. ΔOD of 1725 *nm* and 1800 *nm* were correlated to moisture content, with a standard error of 2.1%; ΔOD of 1650 *nm* and 1725 *nm* were used to predict fat content, with a standard error of 1.4%. Geladi et al. (1985)[19] introduced the spectroscopic pre-processing method: multiplicative scatter correction (MSC) based on their research on NIR reflectance on meat samples. The MSC method was then widely used in other NIR applications. Berzaghi et al. (2005)[72] investigated NIR reflectance of breast meat samples of laying hens with different diets. They measured NIR reflectance in a spectral range of 1100 *nm* - 2498 *nm* at every 2 *nm*, then built PLS calibration on multiple physicochemical features, partial least squares discriminant analysis (PLS-DA) classifier on different diet types. Based on the results from 72 investigated meat samples, Berzaghi et al. found that prediction of protein, major fatty acid, DM and lipids were adequately accurate; predictions of lipid oxidation parameters, ash, color and pH value were relatively poor. Classification of the control and the enriched diets was achieved with 100% accuracy. They concluded that NIR reflectance could be used for quality control, prediction of composition and dietary treatments on breast meat. Anderson (2007)[73] evaluated the performance of FOSS foodscan NIR spectrometer on prediction of fat, moisture and protein content in meat products. The artificial neural networks (ANNs) method was deployed to build calibrations. Anderson assessed the within-lab and between-lab reproducibility of the method. They also notified that NIR detection of meat products was proved by AOAC for commercial applications.

There are plenty researches on using NIR spectroscopy for prediction of sensory parameters of meat and meat products, but results show less reliability. Rødbotten et al. (2000)[74] reported on NIR prediction of beef quality in the early post mortem stage. They indicated that correlation coefficiets on intramuscular fat content were relatively high (0.78 - 0.85), however, correlation coefficients for Warner-Bratzler (WB) shear press and tenderness were poor (0.47 - 0.68). They

concluded that NIR might not be a precise tool for the prediction of final tenderness. Chan et al. (2002)[75] investigated the possibility of using NIR reflectance on prediction of different quality parameters of pork meat. They reported that prediction accuracies for major constituents including XYZ tristimulus colour values, moisture, fat and protein were satisfactory. However, calibrations for sensory and technological parameters vary from marginally good to very poor. Andrés et al. (2007)[76] measured NIR reflectance of 232 muscle samples of lamb, with a target of prediction of flavour, juiciness, texture, and overall liking. Results indicated that the correlation between NIR reflectance and sensory parameters were only significant in a low proportion of the variability in the taste panel traits. However, they also indicated that the discrimination between extreme samples could be done by NIR spectroscopy. The result suggested that NIR spectroscopy could be used to classify meat/meat products into high/low quality classes.

2.1.4.5 Non-agricultural products

NIR spectroscopy is widely accepted to various non-agricultural applications, due to its high speed, low cost, non-destructive and relatively high accuracy. For example, NIR spectroscopy has been applied on identification and qualification of pharmaceutical products, including raw materials[77][78][79][80] and tablets[78][81][82]; analysis of physical and chemical properties of polymers[83][84][85][86][87]; biomedical tests including detection on human blood[88][89][90] and tissue[91][92]; compositional analysis of minerals[93] [94][95]; assessment of soil properties[96][97] [98][99][100] and many other real-world applications.

2.2 Principal component regression and partial least squares regression

2.2.1 Principal component regression

The target of PCA is to transform the variables from the original matrix $\mathbf{X}$, with columns of $\mathbf{x}_i (i = 1, \ldots, I)$ into a lower dimensional matrix $\hat{\mathbf{T}}$, with columns of $\hat{\mathbf{t}}_a (a = 1, \ldots, A)$, where $A \leq I$. Instances in $\hat{\mathbf{T}}$ are called principal components of

$\mathbf{X}$. Notice that all variables in $\mathbf{X}$ are mean-centered by default. In order to get the principal components, a set of orthogonal and normalized loading vectors $\hat{\mathbf{P}}$, with columns of $\hat{\mathbf{p}}_a (a = 1, \ldots, A)$ are calculated such that:

$$\hat{\mathbf{t}}_a = \mathbf{X}\, \hat{\mathbf{p}}_a. \tag{2.3}$$

$\hat{\mathbf{t}}_1, \ldots, \hat{\mathbf{t}}_A$ are in a descending order w.r.t the variance from $\mathbf{X}$. To achieve this, $\hat{\mathbf{P}}$ can be found by maximizing the corresponding scores. More specifically, for $\hat{\mathbf{p}}_1$ we have:

$$\hat{\mathbf{p}}_1 = \underset{\|\hat{\mathbf{p}}_1\|=1}{\arg\max} \{\| \hat{\mathbf{t}}_1 \|^2\}. \tag{2.4}$$

Substitute $\hat{\mathbf{t}}$ with Equation 2.3:

$$\begin{aligned} \hat{\mathbf{p}}_1 =& \underset{\|\hat{\mathbf{p}}_1\|=1}{\arg\max} \{\| \mathbf{X}\, \hat{\mathbf{p}}_1 \|^2\} \\ =& \underset{\|\hat{\mathbf{p}}_1\|=1}{\arg\max} \{\hat{\mathbf{p}}_1^T \mathbf{X}^T \mathbf{X} \hat{\mathbf{p}}_1\} \\ =& \arg\max \{\frac{\hat{\mathbf{p}}_1^T \mathbf{X}^T \mathbf{X} \hat{\mathbf{p}}_1}{\hat{\mathbf{p}}_1^T \hat{\mathbf{p}}_1}\}. \end{aligned} \tag{2.5}$$

The resulting sum-of-squares of the first principal component score vector, $\hat{\mathbf{t}}_1^T \hat{\mathbf{t}}_1$, is the largest eigenvalue $\hat{\tau}_1$ of the matrix $\mathbf{X}^T \mathbf{X}$, and $\hat{\mathbf{p}}_1$ is the corresponding eigenvector.

Further principal component scores can be calculated by extracting previous principal components from $\mathbf{X}$, and maximizing the variance in the residual. More specifically, residual in $\mathbf{X}$ after removing $k-1$ components, $\hat{\mathbf{X}}_k$, can be expressed as:

$$\hat{\mathbf{X}}_k = \mathbf{X} - \sum_{a=1}^{k-1} \hat{\mathbf{t}}_a\, \hat{\mathbf{p}}_a. \tag{2.6}$$

The k^{th} loading vector $\hat{\mathbf{p}}_k$ can be calculated by:

$$\hat{\mathbf{p}}_k = \arg\max\{\frac{\hat{\mathbf{p}}_k^T \hat{\mathbf{X}}_k^T \hat{\mathbf{X}}_k \hat{\mathbf{p}}_k}{\hat{\mathbf{p}}_k^T \hat{\mathbf{p}}_k}\}. \tag{2.7}$$

Finally the principal component score matrix can be calculated by:

$$\hat{\mathbf{T}} = \mathbf{X}\hat{\mathbf{P}}. \tag{2.8}$$

Remember that both the loading vectors and the principal component scores are orthogonal, i.e. the following constraints are satisfied:

$$\hat{\mathbf{P}}^T \hat{\mathbf{P}} = \mathbf{I}, \tag{2.9}$$

$$\hat{\mathbf{T}}^T \hat{\mathbf{T}} = diag(\hat{\tau}_a). \tag{2.10}$$

In Equation 2.10 $\{\hat{\tau}_a, a = 1, \ldots, A\}$ are eigenvalues of the matrix $\mathbf{X}^T\mathbf{X}$, and the covariance between the score for any two different principal component scores is zero:

$$Cov(\hat{\mathbf{t}}_i, \hat{\mathbf{t}}_j) = \hat{\mathbf{t}}_i^T \hat{\mathbf{t}}_j = 0, \qquad for \quad i \neq j. \tag{2.11}$$

The final target of PCR is to find a regression equation that can be directly applied on $\mathbf{X}$, namely regression coefficients $\hat{b_0}$ and $\hat{\mathbf{b}}$ such that

$$\hat{y}_n = \hat{b}_0 + \mathbf{x}_n \hat{\mathbf{b}}, \tag{2.12}$$

where $\hat{y}_n$ is the n^{th} element of $\mathbf{y}$ and $\mathbf{x}_n$ is the corresponding n^{th} row vector in $\mathbf{X}$. $\hat{\mathbf{b}}$ is a column vector of regression coefficients. In PCR, the regression equation is found by a direct MLR of $\mathbf{y}$ on principal component scores $\hat{\mathbf{T}}$:

$$\hat{y}_n = \hat{q}_0 + \hat{\mathbf{t}}_n \hat{\mathbf{q}}, \tag{2.13}$$

where $\hat{q}_0, \hat{\mathbf{q}}$ are the regression coefficients on PCA scores and $\hat{\mathbf{t}}_n$ is the n^{th} row vector in $\mathbf{T}$. A standard solution to Equation 2.13 is:

$$\hat{\mathbf{q}} = (\hat{\mathbf{T}}^T\hat{\mathbf{T}})^{-1}\hat{\mathbf{T}}^T\mathbf{y}. \tag{2.14}$$

The relationship between $\hat{\mathbf{q}}$ and $\hat{\mathbf{b}}$ can be described by:

$$\hat{\mathbf{b}} = \hat{\mathbf{V}}\hat{\mathbf{q}}, \tag{2.15}$$

where $\hat{\mathbf{V}}$ is the weights matrix. In PCR $\hat{\mathbf{V}} = \hat{\mathbf{P}}$. Combine Equation 2.8, Equation 2.10, Equation 2.14 and Equation 2.15 we can get:

$$\hat{\mathbf{b}} = diag(\frac{1}{\hat{\tau}_a})\hat{\mathbf{P}}\hat{\mathbf{P}}^T\mathbf{X}^T\mathbf{y}, \tag{2.16}$$

$$\hat{b}_0 = \bar{y} - \bar{\mathbf{x}}\hat{\mathbf{b}}, \tag{2.17}$$

where $\bar{\mathbf{x}}$ is the mean row vector of $\mathbf{X}$.

2.2.2 Partial least squares regression

The main strategy behind the PLS regression is very similar to the PCR method. Original inputs in $\mathbf{X}$ are first dimensionally reduced to $\mathbf{T}$, and then MLR is applied to find the regression equation. However, different from PCR, $\mathbf{T}$ is calculated to maximize its covariance with $\mathbf{y}$. If we denote $\mathbf{X}_0$ and $\mathbf{y}_0$ as mean-centered input and reference variables:

$$\mathbf{X}_0 = \mathbf{X} - \mathcal{I}\bar{\mathbf{x}}, \tag{2.18}$$

$$\mathbf{y}_0 = \mathbf{y} - \mathcal{I}\bar{y}, \tag{2.19}$$

where $\mathcal{I}$ is a column vector of ones and $\bar{\mathbf{x}}$ is the mean row vector of $\mathbf{X}$.

The $k^{th}(k > 1)$ PLS component can be found by maximizing covariance between $\mathbf{X}_{k-1}\mathbf{w}_k$ and $\mathbf{y}_{k-1}$. $\mathbf{X}_{k-1}$ and $\mathbf{y}_{k-1}$ are residuals after removing previous $k-1$ PLS components. $\mathbf{W}$ is a normalized loading weight matrix that project residuals in $\mathbf{X}$ onto the space in $\mathbf{T}$ sequentially. To achieve this, we can use a standard LS

solution, i.e. minimizing the residual $\mathbf{e}_{k-1}$ in the following system:

$$\mathbf{y}_{k-1} = \mathbf{X}_{k-1}\mathbf{w}_k + \mathbf{e}_{k-1}. \tag{2.20}$$

Notice that $\mathbf{w}_k$ is subject to the constraint $\mathbf{w}_k^T\mathbf{w}_k = 1$.

A standard solution for $\hat{\mathbf{w}}_k$ is:

$$\begin{aligned}\hat{\mathbf{w}}_k =& \frac{(\mathbf{X}_{k-1}^T\mathbf{X}_{k-1})^{-1}\mathbf{X}_{k-1}^T\mathbf{y}_{k-1}}{\| (\mathbf{X}_{k-1}^T\mathbf{X}_{k-1})^{-1}\mathbf{X}_{k-1}^T\mathbf{y}_{k-1} \|} \\ =&(\mathbf{X}_{k-1}^T\mathbf{X}_{k-1})^{-1}(\mathbf{y}_{k-1}^T\mathbf{X}_{k-1})^{-\frac{1}{2}}(\mathbf{X}_{k-1}^T\mathbf{X}_{k-1})(\mathbf{X}_{k-1}^T\mathbf{y}_{k-1})^{-\frac{1}{2}}\mathbf{X}_{k-1}^T\mathbf{y}_{k-1} \\ =&(\mathbf{y}_{k-1}^T\mathbf{X}_{k-1}\mathbf{X}_{k-1}^T\mathbf{y}_{k-1})^{-\frac{1}{2}}\mathbf{X}_{k-1}^T\mathbf{y}_{k-1}.\end{aligned} \tag{2.21}$$

The k^{th} PLS score, $\mathbf{t}_k$, can be estimated by projection of $\mathbf{X}_{k-1}$ on $\hat{\mathbf{w}}_k$, i.e.:

$$\hat{\mathbf{t}}_k = \hat{\mathbf{X}}_{k-1}\hat{\mathbf{w}}_k. \tag{2.22}$$

The next residual term $\mathbf{X}_k$ can be found by subtracting the reconstruction term from $\mathbf{X}_{k-1}$:

$$\mathbf{X}_{k-1} = \hat{\mathbf{t}}_k\mathbf{p}_k^T + \mathbf{X}_k, \tag{2.23}$$

where $\mathbf{p}_k$ is the k^{th} X-loading. $\mathbf{p}_k$ can be found by minimizing the residual term, $\mathbf{X}_k$ in Equation 2.23. Again we can use the LS solution to estimate $\mathbf{p}_k^T$:

$$\hat{\mathbf{p}}_k^T = (\hat{\mathbf{t}}_k^T\mathbf{t}_k)^{-1}\hat{\mathbf{t}}_k^T\mathbf{X}_{k-1}. \tag{2.24}$$

By transposing the two sides in Equation 2.24 we can get:

$$\hat{\mathbf{p}}_k = \mathbf{X}_{k-1}^T\hat{\mathbf{t}}_k(\hat{\mathbf{t}}_k^T\hat{\mathbf{t}})^{-1}, \tag{2.25}$$

We can rewrite Equation 2.23 as:

$$\mathbf{X}_k = \mathbf{X}_{k-1} - \hat{\mathbf{t}}_k\hat{\mathbf{p}}_k^T. \tag{2.26}$$

If we apply the same rule to $\mathbf{y}$ we get:

$$\hat{q}_k = \mathbf{y}_{k-1}^T \hat{\mathbf{t}}_k (\hat{\mathbf{t}}_k^T \hat{\mathbf{t}}_k)^{-1}, \tag{2.27}$$

$$\mathbf{y}_k = \mathbf{y}_{k-1} - \hat{\mathbf{t}}_k \hat{q}_k. \tag{2.28}$$

In the PLS algorithm, the loading weights $\hat{\mathbf{W}}$ and the scores $\hat{\mathbf{T}}$ are orthogonal. However, the loadings $\hat{\mathbf{P}}$ are normally non-orthogonal. In order to directly transform the original $\mathbf{X}$ to $\mathbf{T}$, we introduce the weights matrix $\mathbf{V}$. Hence we have:

$$\mathbf{T} = \mathbf{XV}. \tag{2.29}$$

The relationship between weights, loadings and loading weights can be formulated as:

$$\hat{\mathbf{V}} = \hat{\mathbf{W}} (\hat{\mathbf{P}}^T \hat{\mathbf{W}})^{-1}. \tag{2.30}$$

Finally, in order to perform rapid predictions from $\mathbf{X}$, we would like to describe the model by a linear regression equation:

$$\hat{y}_n = \hat{b}_0 + \mathbf{x}_n \mathbf{b}. \tag{2.31}$$

Recall that the relationship between $\mathbf{y}$, $\mathbf{q}$ and $\mathbf{T}$ can be expressed as:

$$\begin{aligned} \hat{\mathbf{y}} =& b_0 + \sum_{k=1}^{A} \hat{\mathbf{t}}_k \hat{q}_k \\ =& b_0 + \hat{\mathbf{T}} \hat{\mathbf{q}}. \end{aligned} \tag{2.32}$$

where A is the maximum number of PLS factors.

Combine Equation 2.15, Equation 2.30 and Equation 2.32 we can express $\hat{\mathbf{b}}$ and b_0 as:

$$\hat{\mathbf{b}} = \hat{\mathbf{W}} (\hat{\mathbf{P}}^T \hat{\mathbf{W}})^{-1} \hat{\mathbf{q}}, \tag{2.33}$$

$$\hat{b}_0 = \bar{y} - \bar{\mathbf{x}}\hat{\mathbf{b}}. \tag{2.34}$$

Both PLS and PCA can be used to reduce the number of variables in spectrum. In general, in order to achieve a similar calibration accuracy, less components are needed in PLS than PCR. However, it does not always mean that the PLS model is less noisy than the PCR model, because the noise level in PLS components are higher than that in the same number of PCR components, due to the supervised dimension reduction procedure of PLS.

2.2.3 Classification methods

PLS regression and PCR are used to build regression models for NIR datasets. Remember that NIR can be used on discrimination of samples. Popular multivariate classification methods include linear discriminant analysis (LDA), quadratic discriminant analysis (QDA), support vector machines (SVM), K-nearest neighbor (KNN), regularized discriminant analysis (RDA), soft independent modeling of class analogy (SIMCA) etc. These techniques are widely used in NIR classification tasks, but are generally built on top of PLS or PCA dimension reduction. A great deal of research has been conducted on applications with such kind of classification techniques on NIR datasets[101][102][103][104].

In this thesis, our study only focuses on regression problems. However, it should be notified that our proposed regression techniques can be easily extended to a classification framework.

2.3 Model evaluation

2.3.1 Error metrics

For a regression problem, error indicates the performance of a predictive model. However, there are various definitions of error. Popular error metrics include precision, accuracy and bias. They are defined as follows:

Accuracy , is equivalent to the root mean squared error (RMSE). Accuracy takes into consideration of both random and systematic errors, i.e. it computes the difference between the predictions and the ground truth. For a statistical interpretation it can be considered that 2 times the RMSE indicates a 95% confidence intervals for the ground truth of the unknown observation.

Bias , also known as systematic errors, measures the average difference between the predicted value and the ground truth. In general, a proper predictor should be bias free, unless the same prediction model is used across different systems or on different types of samples.

Precision , which is equivalent to standard error of calibration (SEC) or standard error of prediction (SEP), only measures random errors.The SEP squared is can be approximated by the RMSEP squared minus the bias squared. In particular, when bias is very close to zero (which is true in most practical cases), RMSEP is very similar to SEP.

Accuracy and precision are frequently used in the assessment of the predictive model. In general we are trying to make predictions as accurate as possible, so accuracy (RMSE) is widely used. However, in some cases systematic errors can be easily corrected, which means precision (SEP) tells the true predictive capability of the model.

2.3.2 Training, validation and test sets

For the studies presented in this thesis, we have three different types of datasets. They have different functionality and serve for different purpose. There are various ways of defining datasets in different literatures, and in our research the definitions are as follows:

Training set is purely for training purpose. Notice that the training process does not include choosing the architecture of the model (number of components in PCR and PLSR, neural network structure, etc.). The training set is mainly used to optimize weights, namely parameters in the regression equations. In addition, selection of pretreatment methods also depends on the training set. The optimal pretreatment methods are obtained by using both the training set and the validation set.

Validation set (optional) is used for choosing the spectral pretreatment methods, the architecture of the model and tuning hyperparameters in some methods. For example, in PLSR and PCR, we can vary the pretreatment methods and the number of components used on the training set. The models obtained are then compared on the validation set. The model with the smallest root mean squares error of validation (RMSEV) is chosen as the final model for practical prediction. In the ideal case, the validation set should be a separate dataset from the training set. However, this is not always available. The dataset often comes as a whole one, and we usually manually split the dataset into training and validation subsets. The procedure is often repeated for a couple of times, and the samples in the training set and the validation set are exchanged during the process. The technique is called cross-validation (CV) and the averaged CV error (RMSECV) is used as an indicator of the performance of the model. In our study, we frequently use CV for model selection and optimization.

Test set is a completely separate dataset. The test set should be kept away from any training or model selection process. The final models obtained are compared and assessed on the test set, to estimate the true prediction performance.

In our study we used five datasets (refer to Chapter 3 for details), four of them have separate training set and test set. The samples in the training set and the test set were collected separately, to avoid any accidental relationship between the samples. One dataset (dataset 1) only has a single group of unsegmented samples. For that dataset we first manually divide the samples into a training set and a test set,

then perform our study. We will talk more about the design of experiment in the corresponding chapter.

2.3.3 Comparing estimates of prediction error

When two different calibration models are given, we need to know how significant is the difference between the predictive performances of the two models. Suppose the two methods have been trained to predict **y** from **X** on the same training set. The two models obtained are then evaluated on a common test set. To summarize the results we generally use the mean of prediction errors (bias), standard deviation of the errors (SEP) and the root mean square error of prediction (RMSEP) as indicators of performance. They are defined as:

$$m = \frac{1}{n}\sum_{i=1}^{n} e_i, \tag{2.35}$$

$$SEP = \sqrt{\frac{\sum_{i=1}^{n}(e_i - m)^2}{n-1}}, \tag{2.36}$$

$$RMSEP = \sqrt{\frac{\sum_{i=1}^{n} e_i^2}{n}}. \tag{2.37}$$

Here we denote m as the mean of prediction errors, s as the standard deviation, n as the number of samples in the test set, and $e_{i,j}$ as the prediction error on the i^{th} sample in the test set for method j. If we denote d as the difference between the errors of the two models:

$$d_{i,j} = e_{i,1} - e_{i,2}, \tag{2.38}$$

and $\bar{d}$ as the mean of the differences:

$$\bar{d} = \frac{1}{n}\sum_{i=1}^{n} d_i, \tag{2.39}$$

then we have the error of $\bar{d}$:

$$s_d = \sqrt{\frac{\sum_{i=1}^{n}(d_i - \bar{d})^2}{n(n-1)}}. \tag{2.40}$$

Hence we can calculate the 95% confidence interval on the difference between the two biases:

$$CI_{bias} = \{(m_1 - m_2) - t_{n-1,0.025} \times s_d, (m_1 - m_2) + t_{n-1,0.025} \times s_d\}, \tag{2.41}$$

where $t_{n-1.0.025}$ is the upper 2.5% point of a t distribution on $n-1$ degrees of freedom. If such a confidence interval includes zero then there is no significant difference between the two biases. Normally for a good predictor the bias should be very small, and the more critical comparison is between the standard deviations, i.e. the SEPs of the two models. The calculations are described as follows:

First we calculate the correlation coefficient between the two sets of prediction errors $e_{i,1}$ and $e_{i,2}$, denote it as r. Then calculate:

$$K = 1 + \frac{2(1-r^2)t^2_{n-2,0.025}}{n-2}, \tag{2.42}$$

and

$$L = \sqrt{K + \sqrt{(K^2 - 1)}}. \tag{2.43}$$

Finally we can find a 95% confidence interval for the ratio on the two SEP values:

$$CI_{SEP} = \frac{SEP_1}{SEP_2} \times \frac{1}{L}, \frac{SEP_1}{SEP_2} \times \frac{L}{1}. \tag{2.44}$$

If such a confidence interval includes one, then there is no strong evidence for a significant difference between the SEP values of the two models. A more detailed discussion can be found in the book by Næs et al[105].

2.3.4 Regression coefficients for nonlinear models

Some of the regression methods presented in this thesis are not linear or piecewise linear. For example, LS-SVM and GPR are nonlinear, nonparametric methods; when using nonlinear activation functions or nonlinear layers (e.g. convolutional layer is a nonlinear transformation), neural networks will also be nonlinear. In this case we cannot use a single regression coefficient curve to describe the model. The regression formula for a nonlinear method is often not mathematically available or cannot be obtained. We consider this as one of the major sources of resistance to popularize black-box regression methods, e.g. neural networks, for NIR calibration. In this study, we propose a method to visualize the regression coefficients of nonlinear regression models numerically, for a purpose of understanding and interpretation.

The visualization method can be used for any predictor, linear or nonlinear. We treat the predictor as a black box processor that transforms an input spectrum to a single prediction value. We denote the input spectrum as $\mathbf{x}$ of n variables $x_1, x_2, \ldots, x_n$, and the predictor as a function $f(\cdot)$. For a linear model the function can be presented as:

$$f(\mathbf{x}) = \sum_{i=1}^{n} w_i x_i + b. \tag{2.45}$$

For a complex nonlinear predictor, such as neural networks, such a formula cannot be obtained. However, for a specific input spectrum, we can use the numerical method to calculate the local regression coefficients:

$$w_i = \frac{f(x_1, \ldots, x_i + \varepsilon, \ldots, x_n) - f(\mathbf{x})}{\varepsilon}, \tag{2.46}$$

where ε is a very small number such as 10^{-6}. Obviously the regression coefficient curve for a nonlinear predictor depends on the input $\mathbf{x}$. In this study, when plotting the regression coefficients of a nonlinear regression model, we randomly draw a few spectra from the dataset and plot corresponding numerical regression coefficients.

By observing the profile and the variability of the regression coefficients, we can examine the robustness and the nonlinearity of the model.

2.3.5 Noise level of a regression model

In general, when evaluating multiple NIR calibration models, we do not always select the one with the smallest RMSEV or RMSECV, but the most robust model with an adequate performance. For example, in a PLS regression model, if we find using more factors does not significantly improve the prediction accuracy, we usually prefer not to use the extra factors. The same rule applies to other methods. We can easily find a less regularized, more complex model that has very low validation error, but this does not mean it works as well on other datasets. The contribution to the high precision may come from overfitting on the training set. In this case we should be careful about the performance - noise level trade off of the model. Normally users just look at the smoothness of the regression coefficient curve, and judge the noise level of the model subjectively. Here we present a quantitative method based on Fourier transform to evaluate the noise level of a calibration model.

We first use fast Fourier transform (FFT), which can transform a finite equally spaced sequence of data in wavelength domain $\mathbf{x}$ of N variables $x_0, x_2, \ldots, x_{N-1}$ into an equally spaced sequence in frequency domain with the same length $v(v_0, \ldots, v_{N-1})$:

$$\begin{aligned} v_k &= \sum_{n=0}^{N-1} \mathbf{x}_n \cdot \exp(\frac{-i2\pi kn}{N}) \\ &= \sum_{n=0}^{N-1} \mathbf{x}_n [\cos(\frac{2\pi kn}{N}) - i \cdot \sin(\frac{2\pi kn}{N})]. \end{aligned} \tag{2.47}$$

The real and the imaginary parts of the output are the Fourier transforms of the spectrum's even and odd parts, respectively. We simply take the absolute value of the output as the magnitudes of the components.

We will shift the zero-frequency component in Equation 2.47 to the center, so the frequency domain has a region of $[-\frac{1}{2d}, \frac{N-2}{2Nd}]$ if N is even and $[-\frac{(N-1)}{2Nd}, \frac{(N-1)}{2Nd}]$ if N is odd. d is the sampling interval in the input sequence, in our case it is the spectral resolution (5 *nm* for example). When the number of variables is sufficiently large (which is true for most of the NIR datasets), the frequency domain has a range of $[-0.5, 0.5]$ per sampling interval. We can denote the ratio on magnitude of high frequency components (≥ 0.3 for example) in a regression curve as the noise level. Figure 2.5 shows two examples of FFT on regression curve. We can clearly see that the noise level increase with the number of factors employed in the model.

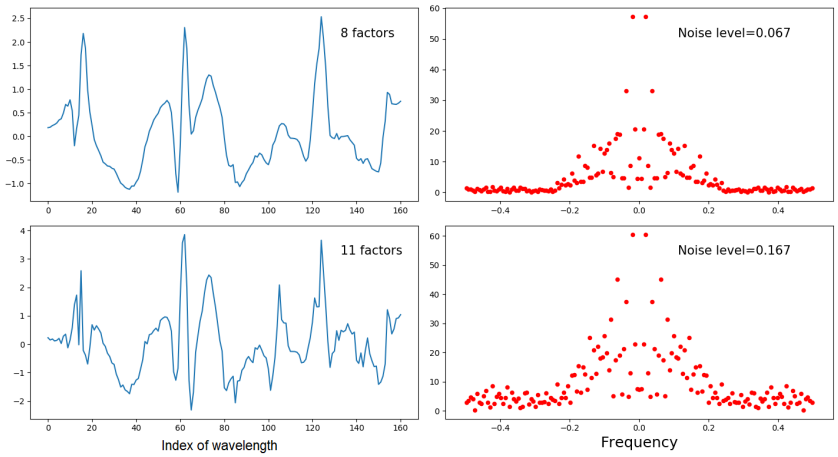


Figure 2.5: Fast Fourier transform on two PLSR models. The simpler model (on the top) has a smooth regression coefficient curve and a lower noise level. The more complex model has more high frequency components in the curve and hence a much higher noise level.

Chapter 3

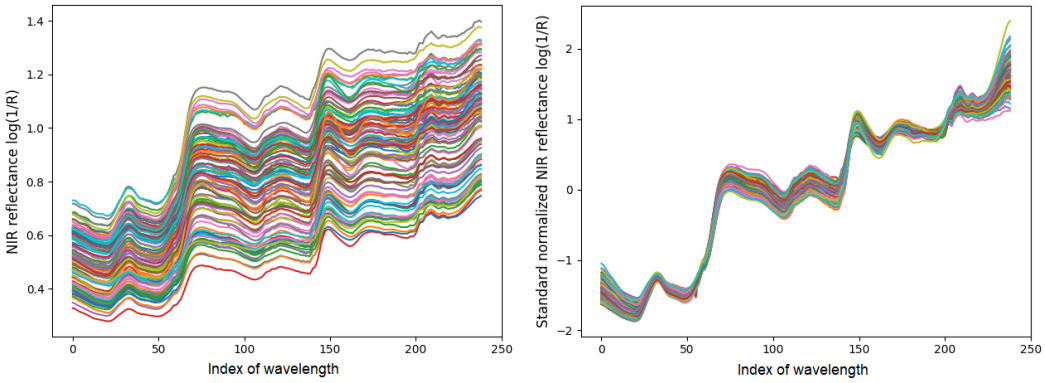
Datasets and software

3.1 Datasets

We use five real-world NIR datasets to evaluate the performances of different calibration methods. The datasets have various sizes and were collected from different sources. Four of them are original to our research, and have been provided by Bühler AG. One dataset is publicly accessible from an on-line database.

Dataset 1 consists of NIR spectra and corresponding nitrogen concentration from 1240 wheat grains. Images were recorded using a MCT hyper-spectral camera and then post processing was used to average the spectral points over the grains to produce a single spectrum per seed. For each spectrum there are 239 valid wavelength readings, spanning from 993.0 *nm* to 2488.4 *nm*. The nitrogen content was then measured using combustion-based chemical analysis. 100 random NIR spectra from dataset 1 are plotted in figure 3.1[1].

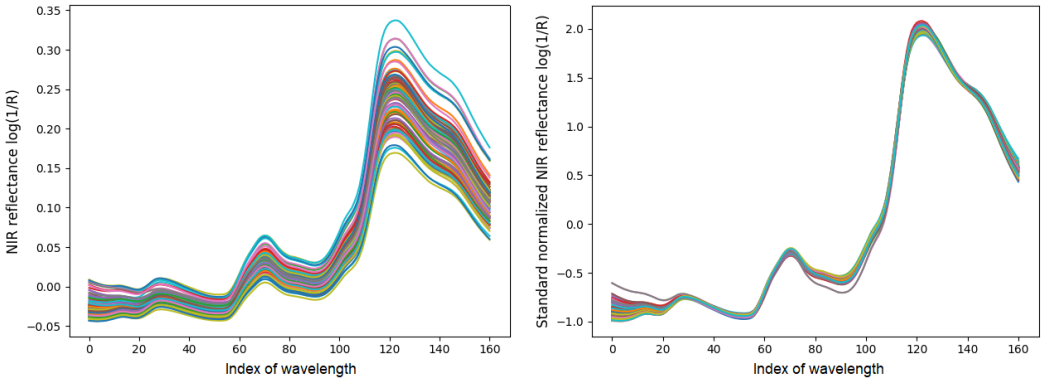
[1]SNV stands for standard normal variate. Here SNV is used purely for visualization purpose. Bias and shift are removed to reveal the key profile of the spectra in datasets.



(a) Raw NIR reflectance spectra. **(b)** NIR reflectance spectra after SNV.

Figure 3.1: Example NIR spectra from dataset 1

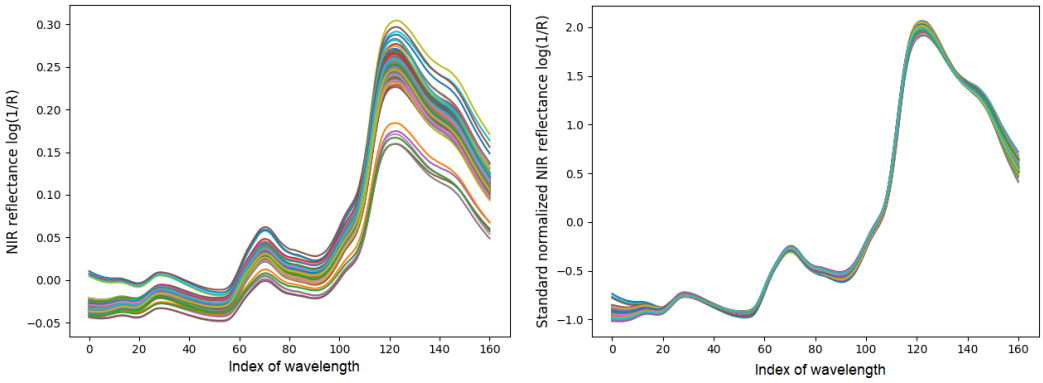
Dataset 2 is on wheat flour. The target constituent is the protein content. Origins of the samples span the globe. The samples also consist of multiple varieties. The dataset contains 1000 training samples and 597 test samples. NIR measurement range is from 850 *nm* to 1650 *nm*, with a spectral resolution of 5 *nm*. 100 random NIR spectra from dataset 2 are plotted in figure 3.2.



(a) Raw NIR reflectance spectra. **(b)** NIR reflectance spectra after SNV.

Figure 3.2: Example NIR spectra from dataset 2

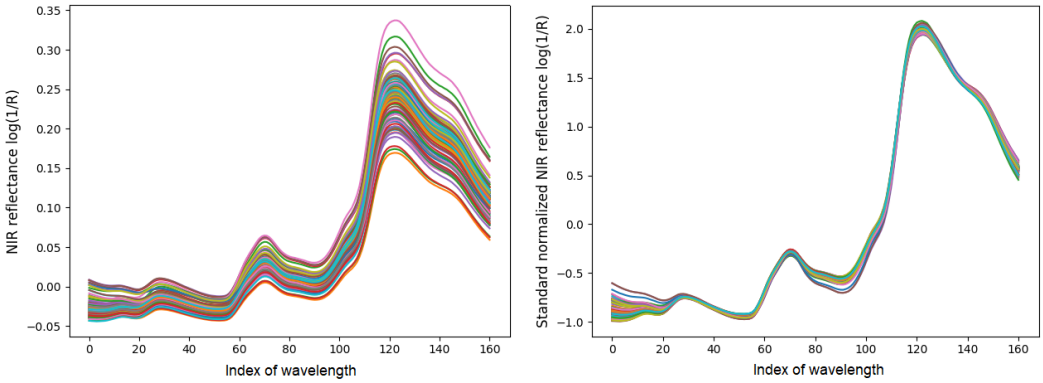
Dataset 3 is also on wheat flour. The target constituent is the ash concentration. The samples have different origins and varieties. There are 6987 training samples and 618 test samples. The spectral range is from 850 *nm* to 1650 *nm*, with a resolution of 5 *nm*. 100 random NIR spectra from dataset 3 are plotted in figure 3.3.



(a) Raw NIR reflectance spectra. **(b)** NIR reflectance spectra after SNV..

Figure 3.3: Example NIR spectra from dataset 3

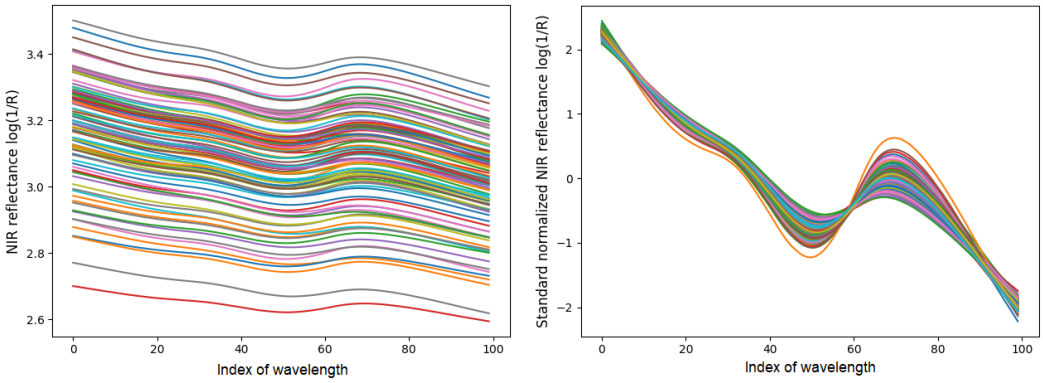
Dataset 4 contains NIR spectra and corresponding moisture contents of wheat flour. Samples have different origins and varieties. The spectral range is from 850 *nm* to 1650 *nm*. Spectral resolution is 5 *nm*. The training set contains 700 samples; the test set has 388 samples. 100 random NIR spectra from dataset 4 are plotted in figure 4.1.



(a) Raw NIR reflectance spectra. **(b)** NIR reflectance spectra after SNV..

Figure 3.4: Example NIR spectra from dataset 4

Dataset 5 is a public dataset[106]. The spectra and the corresponding protein content of single wheat kernels were collected from different locations in Denmark. The dataset contains 415 training samples and 108 test samples. 100 random NIR spectra from dataset 5 are plotted in figure 5.1.



(a) Raw NIR reflectance spectra. **(b)** NIR reflectance spectra after SNV..

Figure 3.5: Example NIR spectra from dataset 5

Datasets 1 - 4 are original to this paper, and are provided by Bühler AG. Datasets 2-4 are on wheat flour. The samples was thoroughly homogenized before being measured. Multiple measurements were repeated on the same sample, under different conditions. There are no obvious outliers in any of the four datasets. The training samples and the test samples were collected and measured independently, under the same spectroscopic and lab systems.

3.2 Software

Here is a list of softwares and development environments used in our studies:

Method	Language	Environment	Main packages
PLSR	Matlab	2014b	Statistics and Machine learning toolbox
PLSR	Python	3.6.1	scikit-learn
PCA	Python	3.6.1	scikit-learn
LS-SVMs	Matlab	2014b	LS-SVMlab
GPR	Matlab	2014b	GPML
HMLR	Python	3.6.1	/
LWR	Python	3.6.1	scikit-learn
CNNs	Python	3.6.1	Tensorflow 1.5.0

Table 3.1: List of Languages, development environments and main packages used in each calibration method.

PLSR in chapter 4 is realized in Matlab in order to compare with LS-SVMs and GPR. PLSR and PCA in Chapters 5-6 are built in Python 3.6.1. The HMLR method

is coded from scratch. We find there is no available package for this algorithm. Tensorflow implemented in chapter 6 was the CPU only version.

Chapter 4

Kernel methods: least squares support vector machines and Gaussian process regression

4.1 Background

The problem of interest is the prediction of target constituent concentration in a sample from its near infrared (NIR) spectrum. Some training samples are used to learn the quantitative relationship between the target constituent and reflectance NIR spectrum. In general, spectral data are highly multivariate and require dimensional reduction. PCR and PLS regression first transform raw data to latent variables, on which MLR is directly applied. Linear methods such as these perform reasonably well in most practical applications. However, in some cases, e.g. when samples are not homogeneous or predictions are required over a large range so that nonlinearity occurs, linear models fail to maintain high accuracy across the range. More specifically, when samples are not homogeneous, the fundamental function that correlates NIR spectrum with the target constituent can vary significantly, which makes it difficult to capture such kind of relationship with a single linear function; Even for homogeneous samples, if the constituent covers a large range, i.e. extreme samples exist, linear regression methods fail to model both extreme samples and normal samples. Nonlinear regression methods are then relevant. In

this chapter we introduce two kernel methods to tackle nonlinearity in NIR calibration: least squares support vector machines (LS-SVM) and Gaussian process regression (GPR).

Support vector machines (SVM) can be used for both classification and function estimation[107]. The method has been used in spectroscopic analysis for over a decade. Demiriz et al. (2001) proposed using SVM as a technique for the prediction of the biological activity of a compound from its chemical structure. They addressed the overfitting issue that arises when there are very many variables and very few training samples. Model selection and bagging strategies were proposed to improve the performance of the model. Xu et al. (2006)[108] introduced SVM as a calibration method in chemometrics for solving classification problems. Devos et al. (2009) further discussed on using SVM in chemometrics, with a focus on cross-validation for parameter optimization. They visualized the SVM models for interpretation purposes. LS-SVM has been introduced as a regression technique[109]. Cogdill and Dardenne (2004)[110] further discussed the LS-SVM regression method in the context of chemometrics and NIR spectroscopy.

Gaussian processes (GP) were introduced for solving regression problems, and are often considered as the state-of-the-art nonparametric Bayesian regression method[111][112]. However, there are very few researches on applying GPR for spectroscopic analysis. Chen et al. (2007)[113] introduced GPR as a regression method for NIR applications, from a perspective of Bayesian modelling. Chen and Wang (2010)[114] continued exploring Bayesian variable selection for GPR, which was reported to improve the predictive capability of the models on NIR applications. However, we find the lack of research into how inference methods can impact GPR on spectroscopic data; Existed studies were carried on relatively small datasets. In this research we are trying to contribute to the knowledge by looking into details on various inference methods in GPR, and extend the study to a large dataset to understand how training set size can be important for PLSR, LS-SVM

and GPR methods.

In this chapter our main focuses are on the following topics:

1) Reintroduce LS-SVM and GPR as regression methods in the context of NIR spectroscopy.

2) Compare the performances of LS-SVM and GPR and the classical regression method of PLSR on our NIR dataset 1 (refer to chapter 3 for detailed dataset description).

3) Address the differences between LS-SVM and GPR from theoretical perspective and on their prediction performances; provide insights on choosing the optimization method and the loss function on practical NIR applications.

The material in this chapter has been published in the Journal of Near Infrared Spectroscopy (2017)[115]

4.2 Methods

4.2.1 Least squares support vector machines

PLSR is normally considered as one of the most powerful high-dimensional linear regression methods. However, the predictive capability of PLSR is sometimes limited due to a non-linear relationship between the NIR spectrum and the chemical constituent of interest. In order to extend the regression method to a non-linear framework, new variables carrying nonlinear information need to be introduced. This can be achieved by mapping the input data into a higher dimensional feature space through a transformation $\phi(.) : \mathbb{R}^k \rightarrow \mathbb{R}^{k_h}$. The mapping process can be done by a linear combination of the newly introduced variables, which includes non-linear transformations and expansion of the original variables. The introduced non-linear feature space potentially has infinite dimensions. Prediction $\hat{y}$ on an observed sample $\mathbf{x}$ is then calculated as follows:

$$\hat{y}(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b, \tag{4.1}$$

where b is the offset term, $\phi(\mathbf{x})$ are the nonlinear features and $\mathbf{w}$ is a vector of regression coefficients. Such a coefficient vector also has potentially infinite dimensions. Given a training set with N measured reference values $\mathbf{y}$ of $y_i, i = 1,2,\ldots,N$ and their NIR spectra with K wavelength readings $\mathbf{X}$ with entries of $x_{ij}, i = 1,2,\ldots,N; j = 1,2,\ldots,K$, we can construct a ridge regression on the nonlinear feature space with a loss function defined as:

$$\mathcal{J} = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \gamma\frac{1}{2}\sum_{i=1}^{N} e_i^2, \tag{4.2}$$

under the constraint:

$$y_i = \mathbf{w}^T\phi(\mathbf{x}_i) + b + e_i, \qquad i = 1,\ldots,N. \tag{4.3}$$

In Equation (4.2) γ determines the regularization strength, which is related to the signal amplitude in the measurements. When the signal is strong we can use large γ, otherwise we should use a relatively smaller γ to prevent overfitting.

To minimize the loss function defined in Equation (4.2), we can construct the Lagrangian with equality constraint[109]:

$$\mathcal{L}(\mathbf{w}, b, \mathbf{e}; \boldsymbol{\alpha}) = \mathcal{J} - \sum_{i=1}^{N} \alpha_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b + e_i - y_i), \tag{4.4}$$

where $\boldsymbol{\alpha}$ are Lagrangian multipliers. The optimal solution can be found by minimizing the Lagrangian function w.r.t. its parameters $(\mathbf{w}, b, e; \boldsymbol{\alpha})$:

$$\begin{aligned}
&\frac{\partial\mathcal{L}}{\partial\mathbf{w}} = 0 \rightarrow \mathbf{w} = \sum_{i=1}^{I} \alpha_i\phi(\mathbf{x}_k),\\
&\frac{\partial\mathcal{L}}{\partial b} = 0 \rightarrow \sum_{i=1}^{I} \alpha_i = 0,\\
&\frac{\partial\mathcal{L}}{\partial\mathbf{e}} = 0 \rightarrow \alpha_i = \gamma e_i,\\
&\frac{\partial\mathcal{L}}{\partial\alpha_i} = 0 \rightarrow \mathbf{w}^T\phi(\mathbf{x}_i) + b + e_i - y_i = 0.
\end{aligned} \tag{4.5}$$

By substituting $\mathbf{w}$ and e_i, the solution for $\boldsymbol{\alpha}$ and b can be expressed as:

$$\begin{bmatrix} 0 & \mathbf{1}^T \\ \mathbf{1} & \boldsymbol{\Omega} + I/\gamma \end{bmatrix} \begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix}, \tag{4.6}$$

where Ω is the kernel function:

$$\begin{aligned} \Omega_{nm} &= \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_m) = \mathcal{K}(\mathbf{x}_n, \mathbf{x}_m), \qquad n, m = 1, 2, \ldots, I. \\ \boldsymbol{\Omega} &= \mathcal{K}(\mathbf{X}, \mathbf{X}). \end{aligned} \tag{4.7}$$

In the following study we only consider using the radial basis function (RBF) kernel, formulated as:

$$\mathcal{K}(\mathbf{x}_n, \mathbf{x}_m) = \exp\left(-\frac{\| \mathbf{x}_n - \mathbf{x}_m \|^2}{l^2}\right). \tag{4.8}$$

The prediction formula for a new observation $\mathbf{x}_{N+1}$ can be expressed as:

$$\begin{aligned} \hat{y}(\mathbf{x}_{N+1}) &= \sum_{i=1}^{N} \alpha_i \phi(\mathbf{x}_{N+1})^T \phi(\mathbf{x}_i) + b \\ &= \sum_{i=1}^{N} \alpha_i \mathcal{K}(\mathbf{x}_{N+1}, \mathbf{x}_i) + b. \end{aligned} \tag{4.9}$$

By substituting $\boldsymbol{\alpha}$ by the solution of Equation (4.6), we can obtain the following equation:

$$\hat{y}(\mathbf{x_{N+1}}) = \mathcal{K}(\mathbf{x}_{N+1}, \mathbf{X}) \left[\mathcal{K}(\mathbf{X}, \mathbf{X}) + \frac{N}{\gamma}\right]^{-1} [\mathbf{y} - b \cdot \mathbf{1}] + b. \tag{4.10}$$

Notice that only two free hyperparameters (γ, l) need to be tuned in the final target formulation. γ is influenced by the signal strength, and l, which is sometimes called length-scale in the input space, determines how sensitive $\mathbf{y}$ is to variances in $\mathbf{X}$. When l is small, $\mathbf{y}$ is very sensitive to small variances in the input space and thus the obtained model can give precise predictions, but with a huge risk of overfitting; when l is large, $\mathbf{y}$ changes slowly, so the predictions are less precise but more robust against random noise in $\mathbf{X}$. These two free parameters can be tuned through cross-validation on the training set to minimize some predefined loss functions, for example, squared error loss as utilized in our research.

4.2.2 Gaussian process regression

A GP is a collection of random variables such that any finite subset exhibits a joint Gaussian distribution[116]. First consider that the measured responses in the training set $\mathbf{y}$ of $y_i, i = 1,2,\ldots,N$ has a ground truth of $\mathbf{f}$ of $f_i, i = 1,2,\ldots,N$, which is also called the latent function. In our case the real constituent concentration in the sample is $\mathbf{f}$, and we can only estimate it through chemical measurement, the result of which is $\mathbf{y}$. In the ideal case the difference between the two terms is a Gaussian white noise ε:

$$y_i = f_i + \varepsilon_i. \tag{4.11}$$

Assume this Gaussian white noise term has a mean of zero and a precision of γ, the conditional probability distribution of the measured responses given the latent function can be described by a Gaussian likelihood function:

$$P(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{f}, \frac{1}{\gamma}\mathbf{I}). \tag{4.12}$$

There are many other possible likelihood functions to describe different types of noise. For simplicity in our study only Gaussian likelihood was considered. The latent function described above is the target GP. A GP is fully defined by its mean function $\mathcal{M}$ and covariance function $\mathcal{K}_{GP}$. Given a set of input spectra $\mathbf{X}$, with rows of $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N$, the latent function $\mathbf{f}$ has a joint Gaussian prior:

$$P(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\mathcal{M}, \mathcal{K}_{GP}(\mathbf{X}, \mathbf{X})), \tag{4.13}$$

where $\mathcal{M}$ is the mean vector with the size of N and $\mathcal{K}_{GP}$ is an $N \times N$ covariance matrix. In this research we used the RBF kernel as the covariance matrix. The $(n,m)^{th}$ entry of $\mathcal{K}_{GP}(\mathbf{X}, \mathbf{X})$ is formulated as:

$$\mathcal{K}_{GP}(\mathbf{x}_n, \mathbf{x}_m) = \sigma_s^2 \mathcal{K}(\mathbf{x}_n, \mathbf{x}_m) = \sigma_s^2 \exp\left(-\frac{\| \mathbf{x}_n - \mathbf{x}_m \|^2}{l^2}\right), \tag{4.14}$$

where σ_s^2 tells the signal strength in the dataset and $(\mathbf{x}_n, \mathbf{x}_m)$ are the $(n,m)^{th}$ samples

in the training set. Normally, for notational and computational convenience, the mean function $\mathcal{M}$ is set to $\mathbf{0}$, and this is appropriate if the responses are mean-centered. So the prior described in Equation (4.13) can be modified as:

$$P(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\mathbf{0}, \mathcal{K}_{GP}(\mathbf{X},\mathbf{X})). \tag{4.15}$$

We can multiply the new prior term described by Equation (4.15) with the likelihood function in Equation (4.12), and integrate out $\mathbf{f}$. As a result we get:

$$P(\mathbf{y}|\mathbf{X}) = \mathcal{N}(\mathbf{0}, \mathcal{K}_{GP}(\mathbf{X},\mathbf{X}) + \frac{1}{\gamma}\mathbf{I}). \tag{4.16}$$

We call the probability distribution $P(\mathbf{y}|\mathbf{X})$ the marginal likelihood (also termed evidence). Marginal likelihood is often used as the loss function to tune the three hyperparameters in GP: γ, σ_s^2 and l. Marginal likelihood can be expressed as the integral of the likelihood and the prior over the underlying function $\mathbf{f}$:

$$P(\mathbf{y}|\mathbf{X}) = \int_{\mathbf{f}} P(\mathbf{y}|\mathbf{f})P(\mathbf{f}|\mathbf{X}), \tag{4.17}$$

which can be further expanded as:

$$\begin{aligned} P(\mathbf{y}|\mathbf{X}) =& (2\pi)^{-I/2}|\mathcal{K}_{GP}(\mathbf{X},\mathbf{X}) + \mathbf{I}/\gamma|^{-1/2} \\ & \times \exp(-\frac{1}{2}\mathbf{y}^T(\mathcal{K}_{GP}(\mathbf{X},\mathbf{X}) + \mathbf{I}/\gamma)^{-1}\mathbf{y}). \end{aligned} \tag{4.18}$$

The first term on the right hand side of Equation (4.18): $|\mathcal{K}_{GP}(\mathbf{X},\mathbf{X}) + \mathbf{I}/\gamma|^{-1/2}$ can be interpreted as the regularization term. It penalizes on the complexity of the kernel function. The second term: $\exp(-\frac{1}{2}\mathbf{y}^T(\mathcal{K}_{GP}(\mathbf{X},\mathbf{X}) + \mathbf{I}/\gamma)^{-1}\mathbf{y})$ is the data fitting term. So the marginal likelihood as a loss function for training purpose is a regularized fitting error loss, just like L2 regularization in LS-SVM. The three free hyperparameters γ, σ_s^2 and l in Equation (4.18) can be tuned to maximize this marginal likelihood on the training set. This technique is sometimes called evidence maximization. Unlike cross-validation in PLSR and LS-SVM, evidence maximization evaluates the gradient of the marginal likelihood over hyperparameters, and uses the whole training set at one time, without dividing it into training

and validation subsets. More importantly, the optimization target is not to minimize the squared error loss, which only depends on the point predictions. The likelihood is determined by both the mean and the variance of the predictive distribution.Conceptually, the final target is to obtain a precise and confident predictive model.

It is worth noting that we can also use cross-validation to optimize the hyperparameters in GPR. In addition, we can also set RMSE as the loss function. However, in that case we would lose the information on predictive variance, and the GPR model obtained is then equivalent to a LS-SVM model.

In the context of NIR calibration, the main interest is making predictions on new observations based on the training set $\mathbf{X} = (\mathbf{x}_i, i = 1,2,\ldots,N)$ and $\mathbf{y} = (y_i, i = 1,2,\ldots,N)$. In GPR, to predict on a new NIR spectrum $\mathbf{x}_{N+1}$, we construct a set of $N+1$ latent variables $f_1,\ldots,f_I,f_{N+1}$ with a joint zero mean Gaussian distribution and a predefined covariance function:

$$P(f_1,f_2,\ldots,f_N,f_{N+1}) = \mathcal{N}\left(\mathbf{0}, \begin{matrix} \mathcal{K}_{GP}(\mathbf{X},\mathbf{X}) & \mathcal{K}_{GP}(\mathbf{X},\mathbf{x}_{N+1}) \\ \mathcal{K}_{GP}(\mathbf{x}_{N+1},\mathbf{X}) & \sigma_s^2 \end{matrix}\right). \tag{4.19}$$

Taking into account the Gaussian noise ε in $\mathbf{y}$, the joint Gaussian distribution of $\mathbf{y}$ and f_{N+1} can be expressed as:

$$P(\mathbf{y},f_{N+1}) = \mathcal{N}\left(\mathbf{0}, \begin{matrix} \mathcal{K}_{GP}(\mathbf{X},\mathbf{X}) + \boldsymbol{I}/\gamma & \mathcal{K}_{GP}(\mathbf{X},\mathbf{x}_{N+1}) \\ \mathcal{K}_{GP}(\mathbf{x}_{N+1},\mathbf{X}) & \sigma_s^2 \end{matrix}\right). \tag{4.20}$$

Conditional on $(\mathbf{X},\mathbf{y},\mathbf{x}_{N+1})$ the variable f_{N+1} has the mean and variance of :

$$\begin{aligned} E(\hat{f}_{N+1}|\mathbf{X},\mathbf{y},\mathbf{x}_{N+1}) &= \mathcal{K}_{GP}(\mathbf{x}_{N+1},\mathbf{X})[\mathcal{K}_{GP}(\mathbf{X},\mathbf{X}) + \boldsymbol{I}/\gamma]^{-1}\mathbf{y}, \\ var(\hat{f}_{N+1}|\mathbf{X},\mathbf{y},\mathbf{x}_{N+1}) &= \sigma_s^2 - \mathcal{K}_{GP}(\mathbf{x}_{N+1},\mathbf{X})[\mathcal{K}_{GP}(\mathbf{X},\mathbf{X}) + \boldsymbol{I}/\gamma]^{-1}\mathcal{K}_{GP}(\mathbf{X},\mathbf{x}_{N+1}) \end{aligned} \tag{4.21}$$

GPR gives full probabilistic predictive distribution for the new observations. The predictive variance is sometimes very useful, for example, for outlier detection: we can simply reject predictions with high predictive variance. However, since interpreting variance on predictions requires one more degree of freedom (GPR has one more hyper-parameter), it also increases the difficulty of tuning the hyper-parameters, especially when the training set is a small one.

4.3 Experiments

Three high-dimensional regression methods (PLSR, LS-SVM and GPR) were applied to the same NIR dataset: dataset 1 as described in chapter 3. Their performances on the test set were compared. All of the calculations were performed in Matlab R2014b (The MathWorks Inc, MA, USA). PLSR was achieved with the Statistics and Machine Learning Toolbox; LS-SVM was realized with LS-SVMlab package[117] and GPR was done with GPML package[118].

4.3.1 Preprocessing

Several preprocessing methods were evaluated through leave one out cross validation (LOO-CV) on the PLSR method. In this study, we are trying to show the superiority of LS-SVM/GPR , hence the preprocessing methods were selected by optimizing the performance of PLSR. As a result, we can conclude that the improvements from nonlinear regression methods are not due to the specific choice of the preprocessing methods. The combination of standard normal variate (SNV) + Savitzky-Golay derivative exhibited the smallest cross-validation error. Hence for all the regression methods, raw spectra were first preprocessed by SNV, to remove constant offset and scatter effect by centering and rescaling individual spectrum. Standard normalized spectra were then transformed by Savitzky-Golay 2 side points (i.e. a window length of 5), 2^{nd} polynomial order fitting first derivative. The same preprocessing treatment was applied for PLSR, LS-SVM and GPR.

4.3.2 Randomization

Since we only have a single dataset, the 1240 samples were first randomly divided into two groups: group A (with 360 samples) as the training set and group B (with 880 samples) as the test set. All of the three regression methods were trained on the same training set. Obtained models were then applied to predict nitrogen concentration in the same test set. A hypothesis test was then applied to tell whether the differences are significant (refer to 2.3.2 for details). The splitting-training-testing process was repeated 5 times in total.

4.4 Result

4.4.1 Comparing global SEP

Comparisons on biases and SEP between the 3 models (refer to section 2.3.2 for details) were performed on 5 separate randomized tests. Results presented in Table (4.1) show SEP and biases for the three methods on each randomized test. Table (4.2) shows the 95% confidence intervals for the true differences in biases. Table (4.3) gives the 95% confidence intervals on the ratios of SEP.

Random split	PLS		LS-SVM		GPR	
	SEP	Bias	SEP	Bias	SEP	Bias
1	0.165	0.013	0.134	0.004	0.131	0.004
2	0.166	0.005	0.133	0.002	0.131	0.004
3	0.162	0.01	0.131	0.016	0.123	0.007
4	0.164	0.006	0.127	0.001	0.125	0.006
5	0.168	-0.01	0.126	-0.004	0.125	-0.002
Average	0.165	0.005	0.130	0.004	0.127	0.004

Table 4.1: SEP and Biases (%) for PLSR, LS-SVM and GPR on each randomized test set

Random split	PLS - LS-SVM	PLS - GPR	LS-SVM - GPR
1	(0.002,0.017)	(-0.002,0.036)	(-0.011,0.025)
2	(-0.003,0.024)	(-0.007,0.04)	(-0.013,0.024)
3	(-0.014,0.002)	(-0.014,0.036)	(0.006,0.040)
4	(-0.003,0.012)	(-0.007,0.03)	(-0.01,0.025)
5	(-0.013,0.002)	(-0.031,0.008)	(-0.023,0.011)
Average	(-0.006,0.011)	(-0.012,0.030)	(-0.010,0.025)

Table 4.2: 95% confidence interval for the true difference in biases (% nitrogen)

Random split	PLS / LS-SVM	PLS / GPR	LS-SVM / GPR
1	(1.171,1.283)	(1.198,1.313)	(0.991,1.045)
2	(1.176,1.302)	(1.201,1.327)	(0.992,1.045)
3	(1.190,1.308)	(1.256,1.370)	(1.021,1.069)
4	(1.236,1.352)	(1.243,1.384)	(0.991,1.029)
5	(1.274,1.392)	(1.294,1.404)	(0.993,1.031)
Average	(1.209,1.327)	(1.243,1.359)	(0.998,1.050)

Table 4.3: 95% confidence interval for the ratio of true SEP (% nitrogen)

From above tables we can obtained the following results and conclusions:

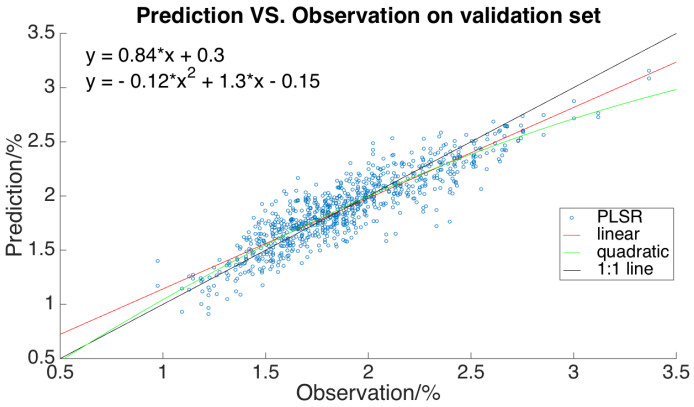
1. For all of the 5 repeated tests, there were consistent and significant differences on the SEP between PLS/LS-SVM and PLS/GPR. The LS-SVM and GPR always out-perform PLSR on predictive precision on our dataset. Averaged ratios of SEP were: PLSR/LS-SVM = 1.270 and PLSR/GPR=1.300.

2. Result from test 3 showed that the SEP and the bias of LS-SVM were statistically larger than GPR, but the differences were both very small. Other tests showed no significant differences on SEP and bias between LS-SVM and GPR. These results indicate that the two non-linear methods were essentially equivalent on predictive capability on this NIR dataset.

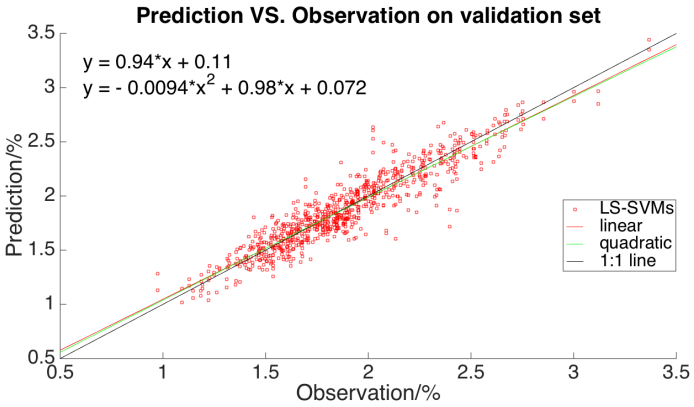
3. Most of the test results showed no significant difference in biases between PLS, LS-SVM and GPR; this is expected since all of the models should be globally unbiased. As a result, significant difference on SEP, which can be interpreted as predictive precision, also indicates significant difference on RMSEP (predictive accuracy).

4.4.2 Local error behavior

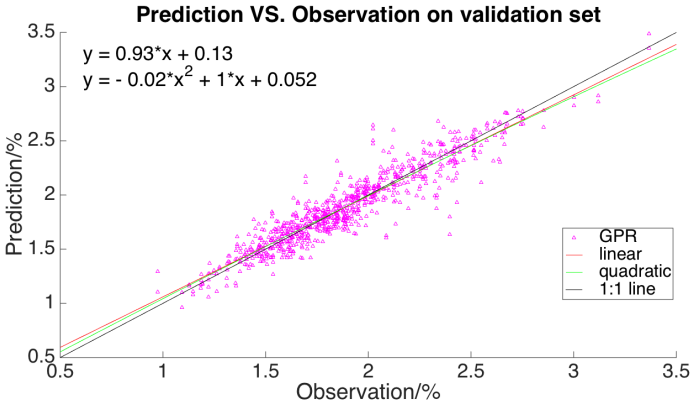
Above results indicate that the two non-linear methods globally perform better than the linear method. In this section, we will explore whether the improvements are consistent over the whole range. One good way to visualize local behavior is to plot predictions against their reference values on the test set, as shown in Figure (4.1).



(a) Predicted vs. Observed for PLSR



(b) Predicted vs. Observed for LS-SVM



(c) Predicted vs. Observed for GPR

Figure 4.1: Predictions for each individual sample in the validation set are plotted against their true observations for PLSR(a) , LS-SVM(b) and GPR(c), with a linear and a quadratic fit to each of them.

Results showed that for PLSR, even through the performance was acceptable in the middle part of range (roughly 1.5%—2.5%), predictions were heavily biased

in the two tails from the regression line, due to two possible reasons:

1) The investigated constituent (nitrogen) concentration has an approximately linear relationship with NIR reflectance within a small range near the population mean, but nonlinear effects become non-ignorable outside this linear range.

2) Predictive capability of PLSR is limited. In order to minimize the global loss, PLSR optimizes the predictions for the samples in the middle part, which accounts for the majority of the population. As a result, within this range, good correlation between predictions and the ground truth can be well maintained. The extreme samples are predicted less well, though this has a limited impact on the overall squared error loss because there are far fewer of them (see Figure 4.1).

In contrast, the non-linear models exhibited fairly consistent standard deviation across the range. Non-linear biases at two tails were removed and predictions were then almost linearly related to the observations for all the samples in the test set. The slope of the regression line was also improved from 0.84 to 0.93 by using GPR. Considering any inverse calibration method implicitly uses population distribution in the training set as a prior, all the predictions are naturally shifted to the sample mean[119], in order to minimize global error. For non-linear models, since the fit was much better, such kind of shrinkage to the mean was much less pronounced.

In order to quantitatively compare predictive performances of PLS, LS-SVM and GPR on the low and high constituent regions, predictions on the first 5% and the last 5% in nitrogen concentration of the samples in the test set were evaluated separately. Again, since all the comparisons were carried on the common subsets, the same hypothesis test method was used to see whether the differences on SEP and bias are significant. Detailed results on comparison between PLSR and GPR are shown in Table 4.4.

Model	LOW(first 5%)		High(last 5%)	
	SEP	Bias	SEP	Bias
PLSR	0.151	0.009	0.213	-0.156
GPR	0.108	0.020	0.160	-0.091
CI	(1.251,1.562)	(-0.035,-0.011)	(1.154,1.521)	(-0.091,-0.024)

Table 4.4: Comparison between PLSR and GPR on different nitrogen ranges. Last row presents 95% confidence interval for the true difference of biases and true ratio of SEP

Some interpretations of these results:

1) It can be seen, unsurprisingly, that GPR is significantly better than PLS over the range. Especially, the improvement is greatest in the two tails.

2) From Table 4.2 we see that there was no significant difference on global biases between PLSR and GPR, and both of them were very close to 0. However, Table 4.4 indicates the significant differences on biases at the two tails, and PLSR was, surprisingly, less biased on the first 5% of the samples. This can be explained by Figure (4.2).

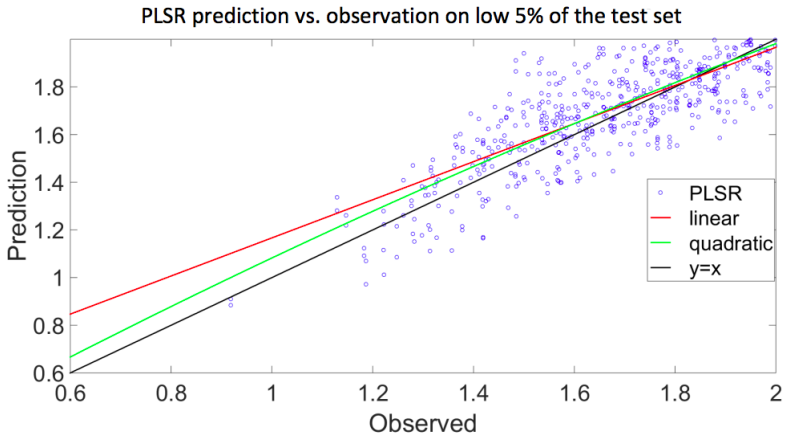


Figure 4.2: PLSR predictions VS. observations on low constituent range. Red:linear fitting; Green: quadratic fitting; Black: target (y=x).

The red line is a linear fit of prediction on observed. The $y = x$ black line shows the shrinkage to the mean that occurs with PLSR. Because this shrinkage is in the same direction on the curvature of this end of the range, it actually reduces the bias

of PLSR. In contrast, at the high end of the range, the shrinkage and curvature are in opposite directions, and PLSR has a very high bias compared to GPR.

4.4.3 Learning efficiency

All of the above conclusions are obtained on a fixed-size training set. A very important question is how efficiently each method learns with a growing training set. In practice, the reference measurement can be very expensive and time consuming. Sometimes there are no sufficient training samples. Then it becomes desirable to understand how these different methods perform with a varying training set size. In Figure (4.3), predictive accuracy on the test set (RMSEP) is plotted against the number of samples in the training set. For each fixed size of training set, RMSEP is averaged over 5 randomized splitting of training and test sets.

Figure 4.3: RMSEP for 3 models with varying training set size. PLSR: Cross-validations was performed to choose the number of factors for each test ; GPR and LS-SVM: RBF kernel.

It can be observed that errors for all the 3 models decrease when the training set size increases, and then gradually converge to their limits. However, they have different learning speeds. LS-SVM and PLSR have similar trends, except for a roughly constant offset on the RMSEP between the two models. For GPR, when the training set size is small, the predictive capability is extremely poor (a RMSEP of 0.38% is close to the standard deviation of the population distribution in the test set so the GPR model has almost 0 predictive capability), but the accuracy improves

quickly with an increasing training set size, and soon becomes non-distinguishable from LS-SVM.

This is the first time in this study when GPR and LS-SVM exhibit significant difference in their predictive performance. It is important that we understand what makes the difference. By comparing Equation (4.10) and Equation (4.21), it can be seen that the calculations for making point predictions in the two methods are very similar. The covariance function used in GPR is $\mathcal{K}_{GP} = \sigma_s^2 \mathcal{K}$, where $\mathcal{K}$ is the same kernel function used in LS-SVM. If we scale $\frac{1}{\gamma}$ with the same multiplier σ_s^2 then this multiplier actually does not affect $E(f_{N+1})$. By further setting the offset term b in LS-SVM to 0, the two prediction formula are mathematically equivalent. This means, with an appropriate choice of hyper-parameters, the two methods can produce identical predictions. However, they have different strategies for hyper-parameter optimization.

1) In LS-SVM, the goal is to minimize the squared error loss by cross-validation, whereas GPR tries to maximize marginal likelihood on the training set. They are very different optimization targets. Imagine that one model gives correct predictions everywhere in the test set, but with quite large variances. Under the criteria of LS-SVM it is a perfect model because it has 0 squared error loss. However, it is a bad predictor for GPR because the predictive variance is too high. The loss function in GPR — marginal likelihood is not strictly equivalent to the predicative capability when we only care about predictive mean. In this sense marginal likelihood might not be suitable to indicate the quality of the model, especially when the training set is a small one, because in that case there is not enough information to estimate the full probability distribution. The squared error loss depends only on the predictive mean, so it concentrates more on giving accurate predictions, which helps to generalize better when the available evidence is weak. However, if the probabilistic information is needed, then we need the predictive variance on new observations. This is also the advantage of GPR. However we

inevitably need more training samples in that case.

2) In our study, LS-SVM used LOO-CV to tune the hyperparameters, but GPR implemented here used evidence maximization (EM). Notice that we can also easily set LOO-CV as the inference method in GPR.Tests showed that different inference methods led to different results. We believe that marginal likelihood involves estimating the probability of the observations given the assumptions of the model, whereas LOO-CV has no prior assumption of the model. As a result, LOO-CV is more robust against model mis-specification[120]. Results also indicate that when using LOO-CV for maximizing marginal likelihood, GPR generalizes better when the training set is a small one, refer to Table (4.5) for a comparison on inference methods.

Index	LOO-CV	Evidence Maximization
1	0.287	0.356
2	0.246	0.337
3	0.241	0.304
4	0.248	0.269
5	0.264	0.311

Table 4.5: RMSEP of GPR with LOO-CV and gradient based evidence maximization for 5 different randomized tests. Training set size= 60. Loss function = marginal likelihood.

It seems that LOO-CV is strongly preferable when the training set is small. When data provided is insufficient, LOO-CV conditions more on the observation and hence uses less knowledge from the prior, which can be mis-specified. However, LOO-CV is not always better than EM in tuning the hyperparameters: because cross-validation conditions too much on observations, there is a higher risk of over-fitting. Since GPR implemented in our study used gradient based evidence maximization, unsurprisingly it had relatively worse performance compared to LS-SVM when the training set is a small one. On the other hand, there was no hyper-parameter initialization in our GPR configuration (where in the LS-SVM, coupled simulated annealing was employed to find suitable start values for hyperparameters), which makes it more vulnerable to bad local minima for small training

sets. Notice this could be a small advantage when training set size is large so local minima are not an issue, because it is more computationally efficient.

4.5 Conclusions

Three models, partial least squares regression, least squares support vector machines and Gaussian process regression were introduced and studied on our dataset 1. Here is a summary of key results and conclusions:

1. Nonlinear methods showed an enhanced prediction performance: the averaged ratio on SEP of PLS/LS-SVM and PLS/GPR were 1.270 and 1.300 correspondingly. The two non-linear models had similar performances and both improved significantly compared to PLSR. There was no strong evidence to indicate any significant difference in biases between the three methods. In PLSR, nonlinear effects led to a large bias and error on prediction of extreme samples. Our study proved that the relatively large error occurring with extreme samples can be corrected by the introduced nonlinear methods.

2. We found that predictive accuracy for all of the three methods grew with the training set size. LS-SVM was strictly better than PLSR, independent of the size of calibration set. GPR had poor prediction performance when the training set is small, due to the characteristic of loss function and optimization algorithm. However, the performance of GPR improved quickly with the training set size, and soon became non-distinguishable from LS-SVM. To fix the issue, we can set the sum of squared errors as the loss function in and use cross-validation to tune the hyperparameters in GPR. By doing so the GPR method exhibited an improved predictive performance on small training sets, which was very similar to the LS-SVM method.

It is worth noting that kernel methods may not be the best option for very large dataset. They are computationally expensive both on training and on prediction. For example, in GPR, evaluation of marginal likelihood involves inversion of the

covariance matrix on the whole training set, which has a computational complexity of $\mathcal{O}(N^3)$, where N is the number of samples in the training set; when tuning the hyperparameters, computation of the derivatives requires additional time of $\mathcal{O}(N^2)$. LOO-CV is even more computationally expensive than the EM + gradient optimization algorithm. More importantly, kernel methods are all non-parametric (i.e., they are not formulated by a set of regression coefficients), which means when making a prediction, we need to evaluate the covariance (e.g., exponential of squared distance for RBF kernel) of the new observation with the whole training set. In our case the training set can easily scale to more than 5000 samples. We found size of a kernel model and computational burden on prediction are not trivial. These limitations are problematic in real world applications, especially for a fast speed on-line system.

Chapter 5

Bayesian graphical model: hierarchical mixture of linear regressions

5.1 Background

Hierarchical mixture of models is a tree-structured scheme for regression and classification. The method was first presented by Jorden and Jacobs(1994)[121]. The architecture is similar to some other well-known algorithms including classification and regression trees (CART)[122] and multivariate adaptive regression splines (MARS)[123]. The input space is segmented into a sequence of regions, with a simple function (sometimes termed an expert) fitted in each of them. A set of gating functions are trained to determine which region the input instances belong to. The regression and the gating coefficients are optimized through maximum likelihood. There are several limitations to the method: it is prone to overfitting due to a large number of free parameters; and singularity may arise when any of the regions contains only a small number of training samples. A further improved tree method is random forest[124]. In a random forest model, there are many weak decision/regression trees. Each tree is trained on different samples and different features. It is proved that random forest can improve prediction precision and is more robust against over-fitting and missing values[124]. However, neither classical

decision/regression tree (CART, MARS) or random forest is our desired solution in spectroscopic calibration, due to the following reasons: 1) decision/regression tree and random forest usually consist of many end models, to make it an overall fairly strong predictor. However, we are looking for a solution to do linear model ensemble where we have strict constraint on model complexity. 2) Model interpretation and visualization are very important for industrial applications, especially when lacking of training/test data. Decision/regression tree and random forest are poor in interpretability. In practical applications, these limitations make them less popular on spectroscopic calibration. Hence in the following research, we do not use CART, MARS or random forest as a benchmark on predictive capability.

Waterhouse, Mackay and Robinson(1996)[125] overcame such limitations by adopting a Bayesian treatment for the mixture of experts. They employed Gaussian distributions over regression and gating parameters to keep the model tractable. Similarly, Ueda and Ghahramani(2002)[126] used the joint distribution over all input and output variables, for tractability. These architectures are either not flexible, or redundant for making predictions.

Bishop and Svensen(2002) [127] offered a full Bayesian treatment for the mixture of experts. The gating functions were in sigmoid-like form, which spoils the tractability of the whole distribution. To handle this, optimization was based on a well-defined lower bound on log marginal likelihood. We find the proposed architecture flexible and it can be easily adopted for NIR applications through minor alterations.

A similar calibration method to the mixture of models in the chemometrics context is locally weighted regression (LWR). The method was presented by Clevenland and Devlin (1988)[128], then introduced to NIR calibration by Næs et al.(1990)[129]. In LWR, when making predictions for unknown observations, we first search for a neighbourhood subspace of the unknown observation in the input

, i.e. spectral, space, where a linear regression is built locally. The inputs to the method are usually compressed features, e.g. principal components or PLS factors. However, the method is non-parametric. It is demanding on the size of the training set, especially when predicting extreme samples. Prediction speed is also relatively slow, since predictions are based on a neighbourhood seeking and recalibration procedure (it can be problematic for high speed on-line systems, just like LS-SVM and GPR introduced in chapter 4). In addition, training a LWR model requires prior knowledge on the number of neighbors, choice of weight function and distance measurement. Such kind of hyperparameters can be sensitive to the characteristics of the available training sets and the target constituents.

In our chapter, we explored the use of Bayesian hierarchical mixture of linear regressions (HMLR) for multivariate spectroscopic analysis. The method is fully parametric. The input space was segmented into subsequence regions by a set of probabilistic gating functions. Considering collinearity in spectral data, we used PLS compression prior to the regression method. As a result, a PLS model was trained on each local region. Finally, the PLS factors in each sub model were unfolded back to the original space (raw spectral data), for the purpose of interpretation and visualization.

In this research, our main contribution to knowledge includes:

1. We propose a novel structure of linear regression ensemble for spectroscopic calibration. We explain in detail how to train the model by using Bayesian DAG and variational inference.

2. We derive equations for parameter update from scratch and show how to optimize the model step by step.

3. We visualize the obtained HMLR models and compare them with the PLS

regression models. We show the improvements of the HMLR method in many aspects. We discuss the gating scheme in the HMLR method and provide insights for model interpretation.

The material in this chapter has been published in Chemometrics and Intelligent Laboratory Systems (2018)[130]

5.2 Methods

5.2.1 Dimension reduction and preprocessing

The task is to train a regression model from a set of N training instances $\{\boldsymbol{x}_n, y_n\}$, where $\boldsymbol{x}_n$ is the original NIR spectrum and y_n is the lab measurement of the target constituent. Preprocessing treatments were applied to the raw spectra before calibration. Preprocessing methods were chosen by CV on the PLS regression method in each dataset. Since NIR spectra are high dimensional and mutually dependent, dimension reduction was applied to the preprocessed spectra to reduce the number of variables. PLS with NIPALS decomposition was used to transform original space $\boldsymbol{x}_n$ into a selection of latent variables $\boldsymbol{\phi}_n$. The number of PLS factors to be used were chosen by CV on a classical PLS regression model.

5.2.2 Overview of the graphical model

The structure of HMLR can be illustrated by figure (5.1). Red nodes in the graph represent a sequence of gating functions, black nodes are the component linear regression models. When a new observation $\boldsymbol{x}_{n+1}$ is given, it flows through the decision tree, each gating node $\boldsymbol{v}_i$ decides whether the current component model M_i should be used to predict the observation. In our case the gating nodes are probabilistic. Outputs of the gating nodes are not binary decisions, but a set of probabilistic weights of the corresponding component models. In the end, predictions from all component models are weighted averaged out by those weights.

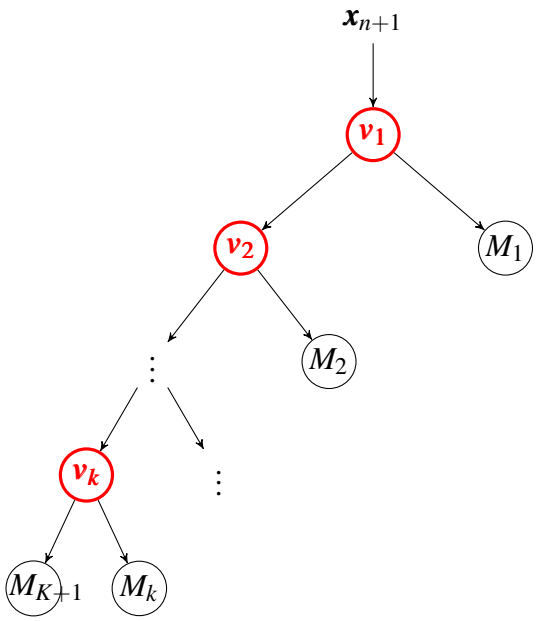


Figure 5.1: Nested structure for integrating linear regression models. Red nodes: gating nodes, determine weights of component linear models; Black nodes: component models, each end model is an independent linear expert

The critical step is to learn the gating functions $\boldsymbol{V}$ and component linear models $\boldsymbol{M}$ from a given training set. For a total number of N training instances $\{\boldsymbol{\phi}_n, y_n\}$, assume there are $K+1$ underlying linear regression functions $\boldsymbol{w}_i, i = 1,2,\ldots,K+1$. Gating functions $\boldsymbol{V}$ determine the weight of each component model for the input $\boldsymbol{\phi}_n$. A label $\boldsymbol{z}_n$ is assigned to each of the training samples. In this study, $\boldsymbol{z}_n$ is a vector of $K+1$ binary variables in training (i.e., either 0 or 1 depending on whether the corresponding end model is assigned). Appropriate regularization was added on these parameters to avoid over-fitting. This is achieved by defining prior distributions on the parameters.

A Bayesian directed acyclic graph (DAG)[131] can be used to describe the dependency of the parameters and the observed data in the model. The DAG is shown in figure (5.2). The rectangular box represents N training samples. The connection from the black dot $\boldsymbol{x}_n$ to $\boldsymbol{\phi}_n$ is a deterministic transformation, which is PLS data shrinkage in this study. $\{\boldsymbol{\alpha}, \boldsymbol{\beta}\}$ are hyperparameters for prior distributions on $\{\boldsymbol{W}, \boldsymbol{V}\}$. $\boldsymbol{\tau}$ is a vector of length of $K+1$, indicates the precision of each component

model.

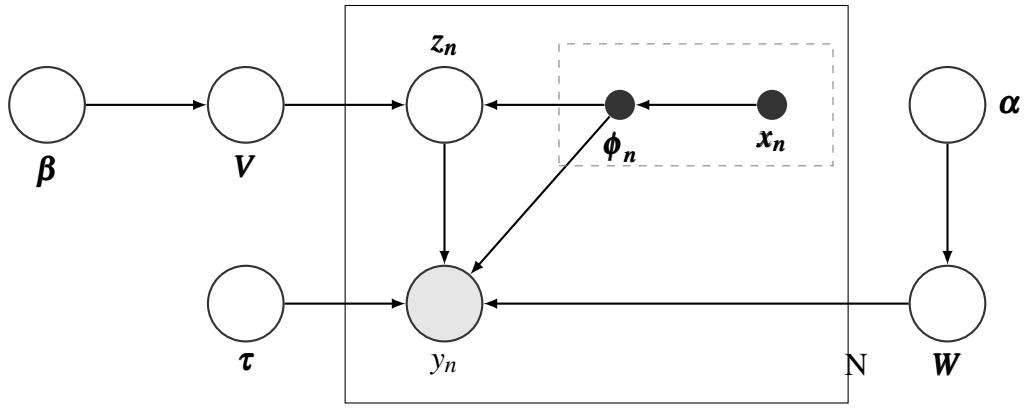


Figure 5.2: Directed graphical model for mixture of linear experts.

5.2.3 Loss function: complete data log-likelihood

As introduced, the target is to find a set of parameters that fits the training set. The complete data log-likelihood (CDLL) is introduced as the loss function. According to the graphical model in figure (5.2), CDLL can be expressed as:

$$p(\{\boldsymbol{\phi_n}, \boldsymbol{z_n}, y_n\}_{n=1,\ldots,N}, \boldsymbol{W}, \boldsymbol{V}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\tau}) = \prod_{n}^{N} \{p(y_n|\boldsymbol{W}, \boldsymbol{z}_n, \boldsymbol{\phi}_n, \boldsymbol{\tau}) p(\boldsymbol{z}_n|\boldsymbol{V}, \boldsymbol{\phi}_n)\} \times p(\boldsymbol{V}|\boldsymbol{\beta}) p(\boldsymbol{W}|\boldsymbol{\alpha}) p(\boldsymbol{\alpha}) p(\boldsymbol{\beta}) p(\boldsymbol{\tau}), \tag{5.1}$$

where $p(y_n|\boldsymbol{z}_n, \boldsymbol{\phi}_n, \boldsymbol{\tau}, \boldsymbol{W})$ is a Gaussian distribution centered at the predictive mean with a precision vector of $\boldsymbol{\tau}$. The gating function $p(\boldsymbol{z_n}|\boldsymbol{V}, \boldsymbol{\phi_n})$ is a sigmoid distribution, which realizes the soft probabilistic splits. The following equations formulate the conditional probabilities:

$$p(y_n|\boldsymbol{z}_n, \boldsymbol{\phi}_n, \boldsymbol{\tau}, \boldsymbol{W}) = \prod_{i=1}^{K+1} \mathcal{N}(y_n|\boldsymbol{w}_i^T \boldsymbol{\phi}_n, \tau_i^{-1} \boldsymbol{I})^{z_{ni}}, \tag{5.2}$$

$$
\begin{aligned}
p(\boldsymbol{z}_n|\boldsymbol{V},\boldsymbol{\phi}_n) &= \prod_{i=1}^{K}\{\sigma(\boldsymbol{v}_i^T\boldsymbol{\phi}_n)^{z_{ni}}[1-\sigma(\boldsymbol{v}_i^T\boldsymbol{\phi}_n)]^{1-z_{ni}}\}^{t(i,\boldsymbol{z}_n)} \\
&= \prod_{i=1}^{K}\{\exp(z_{ni}\boldsymbol{v}_i^T\boldsymbol{\phi}_n)\sigma(-\boldsymbol{v}_i^T\boldsymbol{\phi}_n)\}^{t(i,\boldsymbol{z}_n)}.
\end{aligned} \tag{5.3}
$$

The logistic function $\sigma(x)$ is defined by:

$$
\sigma(x) = \frac{1}{1+\exp(-x)}. \tag{5.4}
$$

Here a binary gating indicator $t(i,z_n)$ is introduced for the training purpose. Notice that in the training procedure, we believe one and only one underlying model is the ground truth for each training sample, and we are trying to maximize the weight on that model. $t(i,z_n)$ indicates whether the i^{th} node is active. For example, if gating node 1 (refer to figure 5.1, the gating node on the top) tells us that the first model is the ground truth, then the rest of the regression tree will not be activated (because there is only one ground truth per sample in the training procedure). $t(i,\boldsymbol{z}_n)$ is formulated as:

$$
t(i,\boldsymbol{z}_n) = \begin{cases} 1 & i=1, \\ 1-\sum_{j=1}^{j<i} z_{nj} & i \geq 2. \end{cases} \tag{5.5}
$$

Notice that Equation (5.3) gives the distribution of z_{n1} to z_{nK}, not including $z_{n(K+1)}$. Since one and only one digit of $\boldsymbol{z}_n$ is 1 (again, because during training we assume there is only one ground truth model), the following condition needs to be satisfied:

$$
p(z_{n(K+1)}=1) = \prod_{i=1}^{K} p(z_{ni}=0). \tag{5.6}
$$

The distributions of the regression coefficients and the gating coefficients are all Gaussian, with precisions from Gamma distributions:

$$
p(\boldsymbol{V}|\boldsymbol{\beta}) = \prod_{i=1}^{K}\mathcal{N}(\boldsymbol{v}_i|0,\beta_i^{-1}\boldsymbol{I}), \tag{5.7}
$$

$$p(\boldsymbol{W}|\boldsymbol{\alpha}) = \prod_{i=1}^{K+1} \mathcal{N}(\boldsymbol{w}_i|\boldsymbol{0}, \alpha_i^{-1}\boldsymbol{I}), \tag{5.8}$$

$$p(\boldsymbol{\alpha}) = \prod_{i=1}^{K+1} Gam(\alpha_i|a_\alpha, b_\alpha), \tag{5.9}$$

$$p(\boldsymbol{\beta}) = \prod_{i=1}^{K} Gam(\beta_i|a_\beta, b_\beta), \tag{5.10}$$

$$p(\boldsymbol{\tau}) = \prod_{i=1}^{K} Gam(\tau_i|a_\tau, b_\tau). \tag{5.11}$$

Now the target is to find a set of $\boldsymbol{Z}, \boldsymbol{V}, \boldsymbol{W}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\tau}$ that maximizes CDLL defined by Equation (5.1) given a fixed training set $\{\boldsymbol{\phi}_n, y_n\}$. The optimization can be achieved by using the expectation maximization (EM) algorithm and variational inference.

5.2.4 Expectation maximization and variational inference

The EM algorithm[132] is often used to find maximum likelihood estimates of the parameters in a statistical model. It runs iteratively to optimize the parameters and the latent variables in the model, where the latent variables are often unobserved. As a result, in EM algorithm the model is segmented into three different parts: observed data set ($\mathcal{X}$), latent variables ($\mathcal{Y}$) and parameters (θ) to make inference on. The EM algorithm has two steps: E-step and M-step. In E-step, latent variables ($\mathcal{Y}$) are estimated according to the observed data ($\mathcal{X}$) and a fixed parameter set (θ), i.e. maximizing the free energy term[133] w.r.t the latent variables $\mathcal{F}(q, \theta)$:

$$\mathcal{F}(q, \theta) = \int q(\mathcal{Y}) \log \frac{p(\mathcal{Y}, \mathcal{X}|\theta)}{q(\mathcal{Y})} d\mathcal{Y}, \tag{5.12}$$

where $q(\mathcal{Y})$ represents any possible distribution of the latent variables. This is proven to be equivalent to minimizing the Kullback-Leibler divergence term[133] $KL[q(\mathcal{Y})||p(\mathcal{Y}|\mathcal{X}, \theta)]$. In M-step, the latent variables are fixed at the results from the E-step as if they are not hidden, and then the likelihood is maximized w.r.t the

parameters. E-step and M-step are iterated until a self-consistent result is obtained.

However, in this study, EM algorithm cannot be directly applied to optimize the model. The main spoiler is the fact that due to the logistic function defined by Equation (5.4), the final CDLL does not belong to the exponential family, which means in E-step, the integral of marginal likelihood is not analytically tractable. At this point, variational inference is introduced to find an approximate solution[134]. The logistic sigmoid function has a lower bound:

$$\sigma(x) \geq \sigma(\varepsilon)\exp\{(x-\varepsilon)/2 - \lambda(\varepsilon)(x^2-\varepsilon^2)\}, \tag{5.13}$$

where

$$\lambda(\varepsilon) = \frac{1}{2\varepsilon}\left[\sigma(\varepsilon) - \frac{1}{2}\right]. \tag{5.14}$$

ε is a new parameter (also termed the variational parameter) introduced by the variational inference, which also needs to be optimized in the training process.

In the variation E-step, the logistic function is substituted by its lower bound, which satisfies the exponential structure. M-step is unchanged. Theoretically, variational inference is not guaranteed to improve the likelihood in every iteration, especially when the approximate lower bound is not very tight. In this study, this is not a severe problem. This will be shown later in the this chapter.

In this study, variational inference was configured as follows: the model was first segmented into three parts: observed data set: $\mathcal{X} = \{\boldsymbol{\phi_n}, y_n\}$; latent variables: $\mathcal{Y} = \{\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\tau}, \boldsymbol{W}, \boldsymbol{V}, \boldsymbol{Z}\}$; variational parameter: $\theta = \{\boldsymbol{\varepsilon}\}$. All the variables as shown in the graphical model were taken as the latent variables and the additional parameter $\boldsymbol{\varepsilon}$ was referred as the parameter to make inference on. The EM algorithm was then employed:

Factored variational E-step: the six latent variables were partitioned into disjoint sets $\mathcal{Y}_s$ as follows:

$$q(\mathcal{Y}) = \prod_{s=1}^{6} q_s(\mathcal{Y}_s). \tag{5.15}$$

Correspondingly, the E-step was also partitioned into iterations: maximize free energy $\mathcal{F}(q,\theta)$ w.r.t $q_s(\mathcal{Y}_s)$ with the other latent variable $q_{j\neq s}(\mathcal{Y}_{j\neq s})$ and the parameter θ fixed, which can be described as:

$$q_s^{new}(\mathcal{Y}_s) := \underset{q_s(\mathcal{Y}_s)}{argmax}\mathcal{F}(q_s(\mathcal{Y}_s)\prod_{j\neq s} q_j(\mathcal{Y}_j), \theta^{old}). \tag{5.16}$$

The solution to Equation (5.16) is:

$$q_s(\mathcal{Y}_s) \propto \exp\langle \log p(\mathcal{X},\mathcal{Y}|\theta^{old})\rangle_{\prod_{j\neq s} q_j(\mathcal{Y}_j)}, \tag{5.17}$$

where notation $<>$ represents the expectation. Notice here it might be a bit difficult to explicitly show the distribution of $q_s(\mathcal{Y}_s)$, but it is not necessary. The only required result is its expectation. Besides, the above formulation only depends on the sufficient statistics of $q_s(\mathcal{Y}_s)$, which significantly facilitates the computational process.

M-Step was unchanged. Free energy $\mathcal{F}(q,\theta)$ was optimized w.r.t the parameter set $\theta = \{\boldsymbol{\varepsilon}\}$, with fixed latent variable distributions (results from Equation 5.17). This can be described by:

$$\theta^{new} = \underset{\theta}{argmax}\mathcal{F}(q^{new}(\mathcal{Y}), \theta), \tag{5.18}$$

which is equivalent[133] to

$$\theta^{new} = \underset{\theta}{argmax}\langle \log p(\mathcal{X},\mathcal{Y}|\theta)\rangle_{q^{new}(\mathcal{Y})}. \tag{5.19}$$

Similarly, it only depends on the sufficient statistics of θ.

All variables in the graphical model were optimized iteratively until a self-consistent result was obtained, i.e. either the CDLL or the parameters themselves converged. Due to the limitation of space, derivations of all the formulations are

attached in Appendix A.

5.2.5 Making the prediction

In this study, a soft mixture of the component models was used to make predictions. For an unknown observation, each component model produced a prediction, and the outputs of the gating functions indicated the weights for each model. Predictions from $K+1$ component models were then weighted to give a single prediction. For all the datasets tested in this study, it was observed that when 2 component models were used the best prediction accuracy was achieved. As a result, K was set to 1 for all the HMLR calibrations.

5.3 Experiments and results

Datasets 3, 4, 5 were used to examine the performance of the algorithm. For all datasets, the raw NIR spectra were preprocessed and then dimensionally reduced by PLS as described above. PLS scores were then passed to MLR, HMLR, and LWR for comparison.

When training the HMLR model, the two component models and the gating function were trained simultaneously. The initial values of $\{\boldsymbol{Z}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\tau}, \boldsymbol{\varepsilon}\}$ were randomly assigned, then the parameters were updated in the order $\boldsymbol{V} \rightarrow \boldsymbol{W} \rightarrow \{\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\tau}\} \rightarrow \boldsymbol{Z} \rightarrow \boldsymbol{\varepsilon}$. The last step (update on $\boldsymbol{\varepsilon}$) corresponds to the M-step, and the others are the factorized variational E-step. Notice that $\boldsymbol{\alpha}, \boldsymbol{\beta}$ should take small initial values (0.1 for example) and a, b in Equation (5.9), Equation (5.10) and Equation (5.11) should also be very small numbers (10^{-4} for example) to make regularization on the regression coefficients weak. In the HMLR model, the raw data are already regularized by the PLS shrinkage. It is not necessary to penalize too much on the coefficients (In fact, there is no penalty on the regression coefficients in PLSR, but they are still robust).

In HMLR, the parameter update process was iterated a maximum of 100 times for each calibration. We recommend checking whether some of the optimization in-

dicators (i.e., CDLL, RMSEC, etc.) are consistent after the whole training process. Since the optimization started at some random values, it is possible the final result is just a local optimum, so the whole optimization process was repeated using several random start values (in this study 80 random initializations were used) to ensure a global optimum for each calibration.

5.3.1 Ash calibration

In this section dataset 3 is studied, refer to chapter 3 for detailed description. CV on PLSR was used to choose the spectral pretreatment and number of PLS factors. We found that the combination of second-order detrend + SNV and 7 PLS factors produced the best PLSR model. We used the same spectral pretreatment and number of PLS factors to train the PLSR, HMLR and LWR models. In what follows we present the detailed training process and some of the key results for this test.

5.3.1.1 Monitoring the training process

The main uncertainty of the HMLR method is that variational inference only uses an approximation to the log-likelihood. In theory, after a whole EM update, the log-likelihood may not increase. We recommend monitoring the optimization process, especially the changes in the CDLL after each update, to make sure the optimization is going in the correct direction. In addition to the CDLL, fitting ability of the model is also relevant. It is essential that after each update the fitting power of the model is also increased (otherwise the improvement on CDLL only comes from regularization), to reject some over-regularized models. Referring to Equation (5.1), only $p(y_n|\boldsymbol{W}, \boldsymbol{z}_n, \boldsymbol{\phi}_n, \boldsymbol{\tau})$ and $p(\boldsymbol{z}_n|\boldsymbol{V}, \boldsymbol{\phi}_n)$ are relevant to the data fitting, the logarithm of the sums of products of the two terms can be taken as the ”fitting score” of the model. Figure (5.3) shows a typical case of the changes in the CDLL and the fitting scores during training.

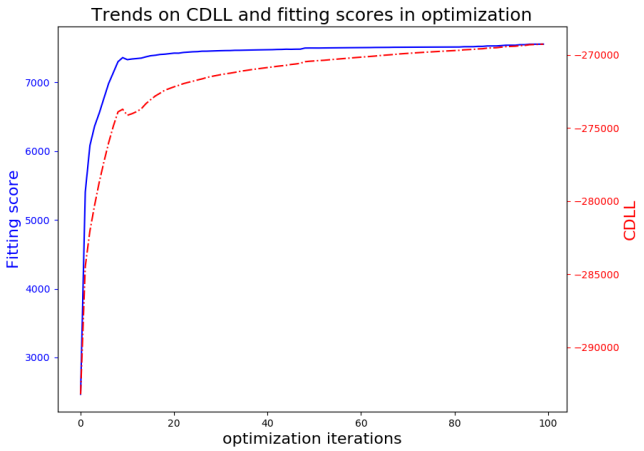


Figure 5.3: Trends in CDLL and fitting scores after each whole EM update. Blue solid line: fitting scores. Red dotted line: CDLL

Figure (5.3) shows that the overall trends in both the fitting score and the CDLL are increasing during optimization. The Fitting score converges after around 20 iterations, while CDLL takes more than 80 iterations to reach self-consistency.

5.3.1.2 Interpretation of the model

As explained, every single sample in the training set was assigned a label z_n, which indicated the cluster of the training sample. This labelling variable was also optimized automatically by the variation inference algorithm. It is useful to show the result on the automatic segmentation found by the algorithm. Figure (5.4) shows the distribution of ash content in the two component models after optimization.

It can be observed that class 1 (blue face colour) comprises the training samples low in ash content, whereas class 2 (orange face color) consists of the samples high in ash content. There is a small overlap, but the two clusters of the samples mainly depends on high and low concentration of ash content.

In the end, the HMLR model is similar to a two-tier PLSR model on ash content. However, the advantage of our proposed approach is the segmentation process is done automatically by the EM and variational inference algorithm, which means it optimizes the data likelihood. Besides, a gating function is obtained simultaneously, which optimizes the segmentation alongside with the component regressions.

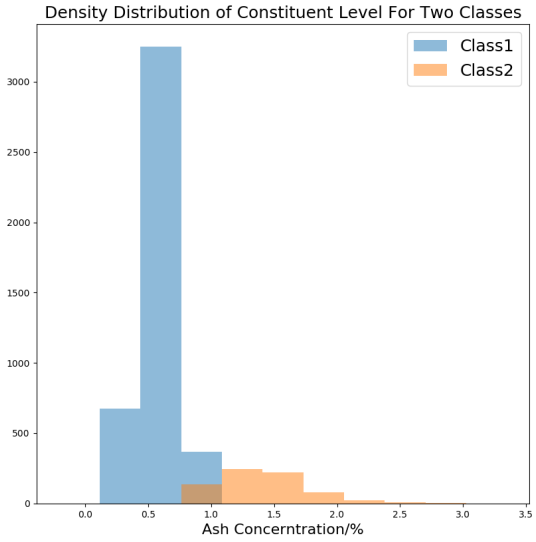


Figure 5.4: Composition of the training samples for the two component models.

Referring to Equation (5.3), the outputs of the logistic gating function represent the probabilities of using the component models, which makes it clear how to combine the component models on prediction.

It is also useful to plot the regression coefficients. Figure (5.5) shows the regression coefficient curves from the PLSR (red dash-dot line) and the two component PLSR models of HMLR (green line and the blue dashed line). The trends of the three coefficient curves are very similar. The signal strengths at some peaks vary slightly. The coefficients from PLSR are closer to that of the component model 1 in HMLR. The phenomenon is expected because most of the training samples were assigned to the component model 1.

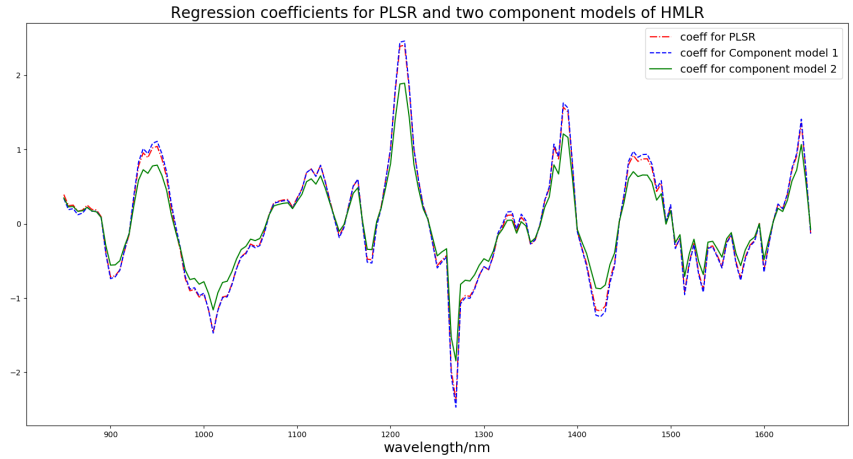


Figure 5.5: Regression coefficients for PLSR and the two component models of HMLR.

Figure (5.6) shows the coefficient curve for the gating function. The trend is very similar to the regression coefficient curve in Figure (5.5). In fact, the segmentation was based on the ash contents so the regression and the gating function have very similar profiles.

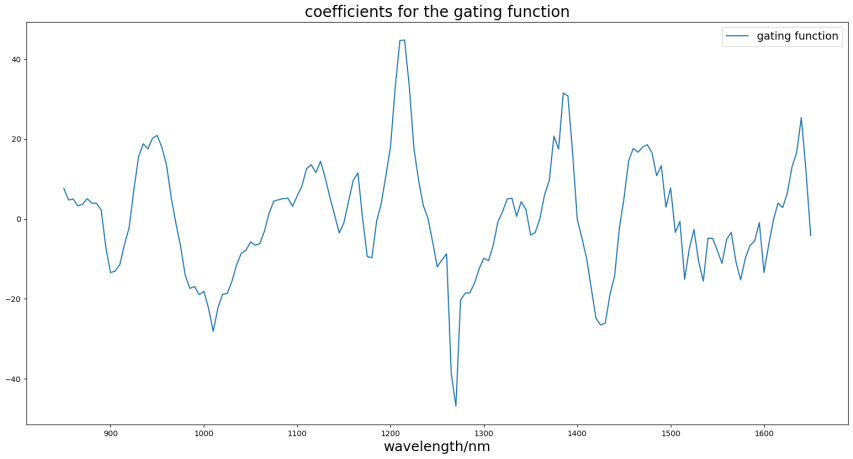


Figure 5.6: Coefficient curve for the gating function

Here is a brief explanation on why the segmentation is based on ash concentration for this dataset. First of all, from a mathematical standpoint, the initial values of the free parameters in the model were randomly assigned. The target of variational inference is to maximise data log-likelihood. So there is no prior preference applied

to the training process on how the segmentation should be done. Considering that the samples used for this study, wheat flour, were thoroughly homogenised before measured, the signal from the same sample is strong and highly reproducible. As a result, the segmentation is based on different types of the intrinsic correlation between the input spectrum $\boldsymbol{X}$ and the target property $\boldsymbol{y}$. Such kind of correlation is slightly different when $\boldsymbol{y}$ varies. The effect was found, by the EM and variational inference, to be the most influential criteria for segmentation on this dataset. Later in another dataset, we will show how the segmentation strategy is different for whole grain samples.

5.3.1.3 Interpretation of τ

Referring to the graphical model described in Figure (5.2), $\boldsymbol{\tau}$ represents the precision of $\boldsymbol{y}$, namely the precision of reference measurement. This value has a physical interpretation behind it, since it is determined by the precision of lab measurement. When training the model, $\boldsymbol{\tau}$ was first assigned randomly, then automatically optimized until self-consistent. It is essential to check whether the final value of $\boldsymbol{\tau}$ is close to the actual precision of the lab measurement. If $\boldsymbol{\tau}$ is too high, then the model is very likely over-fitted; if it is too small, then the model is under-fitted.

Since there are two component models, different τ values were assigned for each of them. Results showed that $\tau_1 = 313$, $\tau_2 = 172$. The equivalent variances on reference measurement of $\boldsymbol{y}$ are 0.003 and 0.006. These values coincide with the known error in the lab measurement. In addition, τ_2 is smaller than τ_1, which is also sensible because the error of lab measurement increases with the ash concentration.

5.3.1.4 Comparison with PLSR and LWR

In this test, the root-mean-square errors of prediction (RMSEP) on the test set for the three models were: PLSR-0.083%; LWR-0.084%; HMLR-0.070%. LWR was based on 13 neighbours (chosen by cross-validation on the training set). The performance of LWR and PLSR are very close. The error obtained by HMLR

was 16% smaller than PLSR. To prove whether the difference is significant, a test for paired predictions was utilized to calculate the true ratio on SEP for the two models. Detailed calculation was introduced in section 2.3.2. The test indicated that the 95% confidence intervals for the true ratio of RMSEP on ash was $\frac{PLSR}{HMLR} \in (1.164, 1.271)$. Since this interval does not include 1, there is a strong evidence that the error of PLSR is significantly larger than the error of HMLR.

Figure (5.7) shows the prediction results from the PLSR model and the HMLR model on the test set. It is messy on the left side, however, for the samples with an ash content above 1%, is is quite apparent that the predictions of HMLR are closer to the reference value.

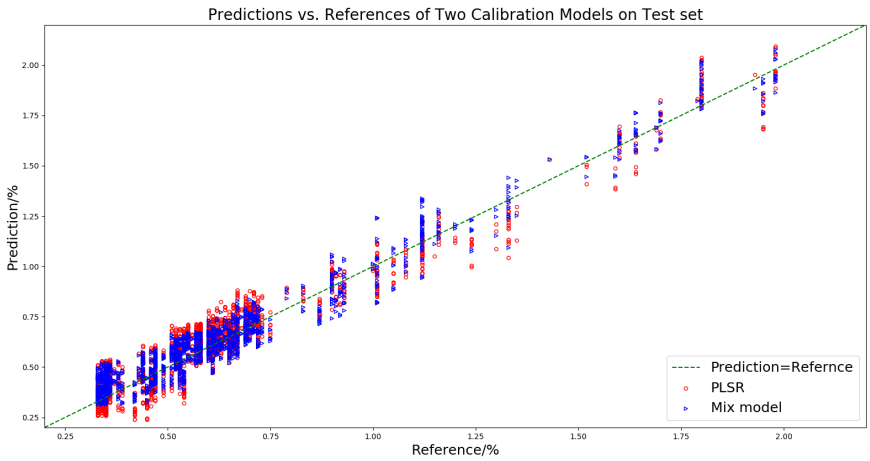
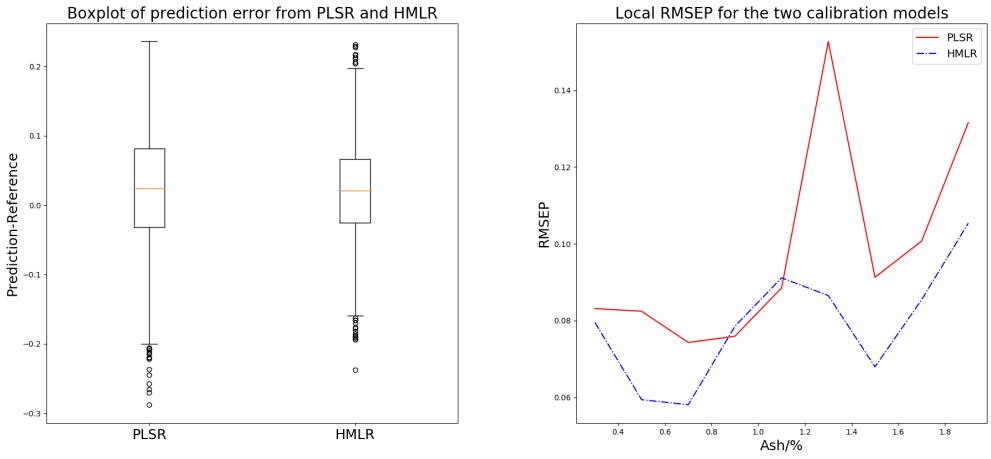


Figure 5.7: Prediction vs. reference on dataset3. Results are from the PLSR model (red circle points) and the HMLR model (blue triangle points).

Some results from the scatter plots in Figure (5.7) are summarized in Figure (5.8). Figure (5.8a) is a boxplot of the prediction errors from the PLSR model and the HMLR model. Median, the first and the third quantile and the inner fences are indicated. It is clear the HMLR model outperforms the PLSR model on global accuracy. Figure (5.8b) illustrates the performances of the two models on different regions. The test set was split into nine local regions with a bin width of 0.2%. The blue dashed line is the accuracy curve of the HMLR model, and the red solid line is for the PLSR model. In most parts of the population, RMSEP from the HMLR

model is smaller than that of the PLSR model, except for the range $0.8\% \sim 1.2\%$, where the two methods have very similar performances. The field is also where the training set is segmented in the HMLR algorithm. So for the samples with ambiguous labels (i.e., the output of the gating function is close to 0.5 for both of the component models), there is no improvement using the HMLR methods. When there is a dominating component model for the prediction, using HMLR can bring significant improvement.



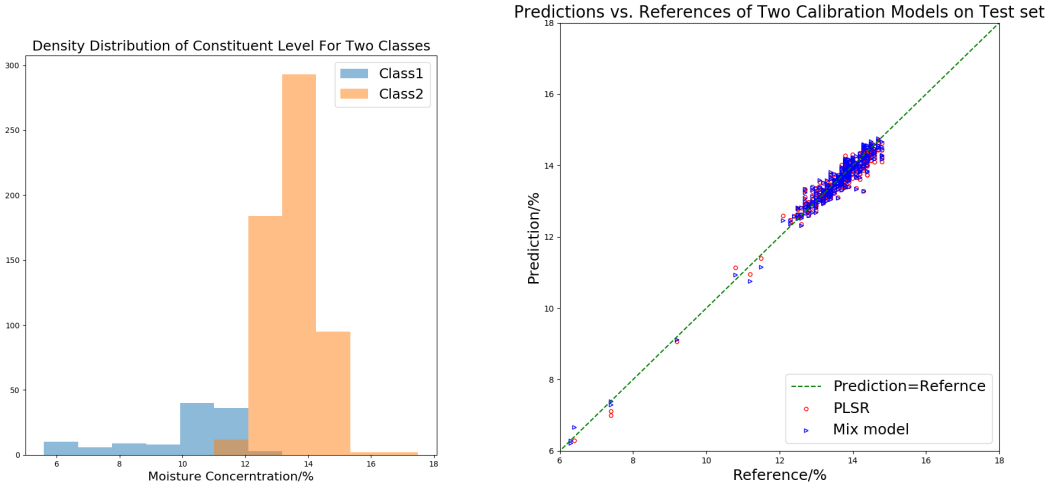
(a) Boxplot of the global errors. **(b)** Local RMSEP for the two models.

Figure 5.8: Comparison of the prediction accuracies of the two ash prediction models

5.3.2 Moisture and protein calibrations

Results on dataset 4 are plotted in Figure (5.9). Dataset 4 is on moisture of wheat flour. From the results it can be observed, similar to the ash calibration, the data is automatically segmented into two subsets according to the moisture content. The boundary of the two classes is at around 12%. Figure (5.9b) shows the predictions on the test set from the two models. Unfortunately, the test samples below 12% are not numerous enough to show the difference between the two models in this region. The RMSEP of the three models on the test set were PLSR:0.228%; LWR:0.222% (8 neighbours); HMLR: 0.212%. There is an improvement in the prediction accuracy by using HMLR, but it is not significant. Practically, calibration on moisture is not challenging. PLSR can produce a satisfactory result (refer to Figure 5.9, predic-

tions from PLSR on extreme samples are not heavily biased or distorted). Moisture has a strong correlation with NIR spectrum, especially for powder samples where signal is robust and reproducible. However the result of the test indicates that even for simple datasets like moisture in flour, the HMLR method does not overfit on the training set. The combination of PLS shrinkage and variational inference is proven to be an appropriate, well-regularised training process. Hence the HMLR method can be used as a routine calibration strategy, even on simple dataset like moisture in flour.

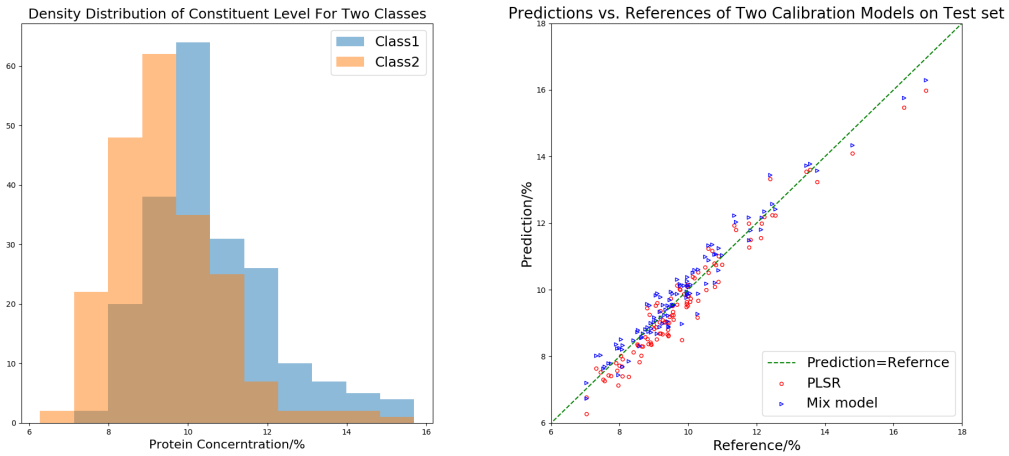


(a) Dataset segmentation on the training set **(b)** Predictions vs. reference from the two models.

Figure 5.9: Segmentation and predictions on dataset 4

Dataset 5 is on protein calibration of whole wheat grains. Data segmentation and prediction results are plotted in Figure (5.10). It can be seen there is more overlap on protein content in the two subsets. The consequence is partly because the distribution of the protein in whole wheat grains is more symmetric than ash and moisture, i.e., the distribution profile has no outstanding asymmetric tails. The other reason is that the gating strategy in this calibration is not entirely based on the protein concentration. Figure (5.11) shows the regression coefficient curve from the PLSR model and the gating function of the HMLR model. It can be seen the two curves partially overlap, mainly on the right-hand side of the spectrum. On the left-

hand side, the gating and the regression functions show different trends. In general, whole grain samples have more complex NIR reflectance spectra than powdered products, and there are other possible segmentation criteria on the training samples. For example, the presentation of the grain (crease side or the opposite side), color, shape and surface conditions.



(a) Dataset segmentation on the training set **(b)** Predictions vs. reference from the two models.

Figure 5.10: Segmentation and predictions on dataset 5

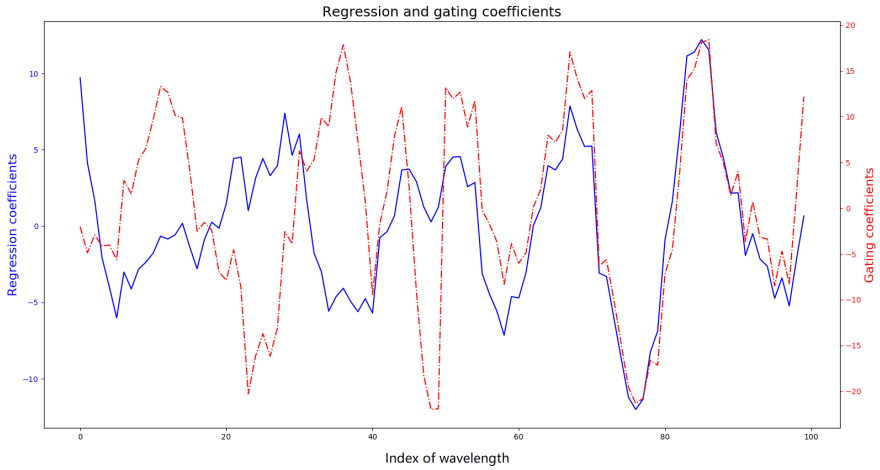


Figure 5.11: PLSR coefficients (6 factors) and gating function coefficients for the protein calibration on dataset 5.

The RMSEP of the three models on the test sets were PLSR:0.425%;

LWR:0.753% (4 neighbors); HMLR:0.386%. The hypothesis test showed that the ratio of SEPs is $\frac{PLSR}{HMLR} \in (1.115, 1.241)$. From Figure (5.10b) it can be seen that the improvement is across the range.

Figure (5.12) shows prediction residual vs. reference of PLSR and HMLR. It can be observed that the bias of the HMLR model is smaller than the PLSR model. The PLSR model had a global bias of -0.210, and that of the HMLR model is 0.008. The method of section 2.3.2 were used to find out the confidence interval for the difference on the biases of the two models. Result indicated 95% confidence interval for the actual difference in bias was $(-0.0267, -0.0229)$. Since this interval does not include 0, there is a strong evidence that the PLSR model has a significantly larger negative bias than the HMLR model.

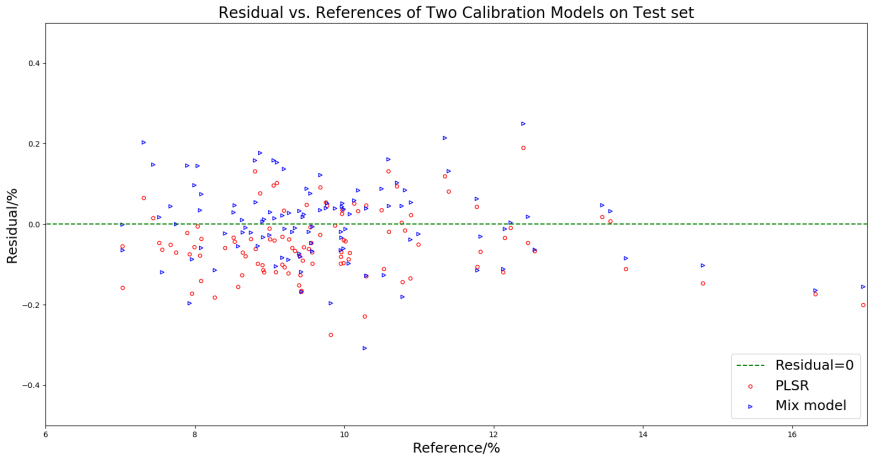


Figure 5.12: Residual vs. reference on dataset 5.

5.4 Conclusion

In this chapter, a new calibration method for NIR spectroscopy, hierarchical mixture of linear regressions, was introduced. The technique automatically searches for a few compositional PLSR models on the training set, along with a set of gating functions to determine the weights of the component models on prediction. Compared to the traditional scheme of ensemble modelling, the HMLR method uses the EM algorithm with variational inference, which makes it entirely automated. No man-

ual data segmentation is needed for training. In addition, the final model obtained is parametric. Similar to the PLSR method, it is interpretable and very compact. In practice, it is possible to understand the model, and monitor the optimization process. It can be implemented efficiently in on-line measurement systems.

The introduced method was tested on three different datasets — ash, moisture and protein content. Investigated samples include wheat flour and whole wheat grains. The HMLR method showed superiority over PLSR and LWR on predictive capability. Based on our experiments, the HMLR method outperforms the PLSR on dataset 3 (ash on wheat flour) and 5 (protein on whole wheat grains); on dataset 4 (moisture content on wheat flour) the two methods have similar performances. As explained, the introduced model can handle challenging datasets, while not over-fitting on easy ones. We recommend using this calibration scheme as a routine method, especially when an interpretable, compact, unbiased and accurate model is desired.

The biggest issue on implementing this method is lack of available software/packages. We built the whole training algorithm from scratch in python. In practical applications, it might be a bit time consuming to develop a software for HMLR calibration.

Chapter 6

Convolutional neural networks for multiple regression

6.1 Backgrounds

Convolutional neural networks (CNN) have recently become one of the most popular solutions for a variety of machine learning tasks, including object detection[135], image classification[136], natural language processing[137], time series classification[138] and many other applications.

Implementation of neural networks (NN) as a chemometrics technique is relatively recent. Long, Gregriou and Gemperline (1990)[139] demonstrated the application of artificial neural networks (ANN) for nonlinear multivariate calibration, based on results on spectroscopic datasets. They reported that the ANN method can tackle nonlinearities and instrument drifts. However, back in the 1990s due to the limitation on computational capability, training a ANN model was tedious and extremely time consuming (typically 5 - 6 hours for 40,000 learning iterations, as reported by Long et al.).

Zupan and Gasteiger(1991)[140] discussed more on the learning procedure of ANN. They recognized ANN approach has too many degrees of freedom, making it impossible to make inference on every parameter in the model. ”Indirect” influ-

ences should be considered as the major tuning procedure in the ANN approach. Such kind of influences include choice of network, architecture, hyperparameters like learning rate and momentum term, selection of training samples and variables.

Wu et al. (1996)[141] discussed on selection of the training set for ANN calibration. Presented results demonstrated that the best training set selection methods are Kennard-Stone and D-optimal, both of them showed greater predictive accuracy than Kohonen self-organized mapping and random selection on NIR datasets.

Boger (2003)[142] presented the method to select wavelengths for spectroscopic calibration based on the analysis of the connection weights of an ANN model. Based on the results presented on a real and an artificial datasets, the method could identify a quasi-optimal small set of input attributes, which gave substantial improvements in prediction accuracy.

There are several nice reviews of applications of ANN on spectroscopic analysis, which comprehensively introduced the mathematics and examples of applications[143][144][145][146].

Two very recent works reported using CNN for spectroscopic analysis. Malek, Melgani and Bazi (2017)[147] explored 1D-CNN for spectroscopic regressions, with integration of particle swarm optimization for the training purpose. Bjerrum, Glahder and Skov (2017)[148] introduced an architecture of CNN + GP for regression tasks, with Bayesian optimization for hyperparameter tuning. However, we found that these works lack interpretation of the CNN models, and the experiments were based on relatively small datasets (up to a few hundred samples). In our research we use a slightly modified 1D-CNN framework, which is proven to be the most efficient structure on our tested datasets; On the other hand, considering computational latency, CNN+GP method is not suitable for our applications. We have discussed the issue in Chapter 4, so the CNN+GP method is not going to be

compared in our research.

In this research, our main contributions to knowledge is:

1. A Feasibility study of using CNN for NIR calibration are carefully done on 3 real-world datasets, with different sizes (varying from 400 to 7000 samples).

2.The predictive capabilities of the CNN models are benchmarked against the classical treatment (PLS regression) with significance tests of difference in performance.

3. Visualization and interpretation of the CNN models are presented, for better understanding and comparison with the classical method of PLSR.

4. We discuss influences of different design choices, model hyperparameters, and provide recommendations and insights for construction CNN models on NIR datasets.

The material of this chapter has been published in Chemometrics and Intelligent Laboratory Systems (2018)[149]

6.2 Methods

6.2.1 Preprocessing and initialization

6.2.1.1 Preprocessing

There are many commonly used preprocessing methods for spectroscopic calibration[105], such as standard normal variate (SNV), detrending, derivative …. In the classical treatment, raw spectra (output of the instruments) are preprocessed by these methods before feeding into a calibration framework. The purpose of spectral preprocessing is to remove instrumental noise, and reduce variance of the spectra from sources such as scatter effect to improve robustness of the calibration model. However, there is no standard rule on choosing preprocessing methods. The common practice is trial-and-error experimentation.

Later in this paper, we will describe how to use the convolutional layer to

automate spectroscopic preprocessing. The only required preprocessing is normalization (i.e. each variable should have a mean of zero and standard deviation of 1 along the column axis). Non-zero-centered data significantly limits the allowed gradient update directions during the optimization process. For example, the worst case is that all inputs into a fully connected layer are positive (or negative). The output of a neuron in such kind of layer will be $f(\sum_i w_i x_i + b)$, where $f(\cdot)$ is the activation function and w_i are the weights. Gradient updates on all weights of such layer can only be either all positive or all negative, which is obviously inefficient.

6.2.1.2 Initialization

Proper initialization of weights in NN ensures fast convergence. If the initial weights are too small, then the signal flowing through the network gradually diminishes and eventually becomes meaningless; if the weights are initialized with too large values, the variance of input data into each layer increases rapidly and soon overflows. We hope that the signal variance does not change significantly when passing through the network. One good idea is to initialize the weights according to the layers' sizes[150]. The weights are initialized by a zero-mean Gaussian distribution whose standard deviation is $\sqrt{\frac{2}{n_l}}$, where n_l is the number of input neurons of the layer. This initialization method helps to keep the variance of all the variables in NN roughly at the same scale.

6.2.2 Layers

A densely connected layer also termed fully connected layer (FC layer), linearly maps input vector into another one. Neurons between two adjacent layers are pairwise connected, refer to Figure 6.1.

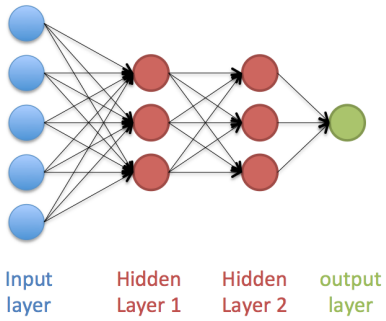


Figure 6.1: A neural network with two hidden layers of 3 neurons each

In our case, the initial inputs are the spectral variables, and the final outputs are the predictions of the target constituent.

Convolutional layer convolves a specific filter applied to the inputs. The filter is slid over the spectra spatially. We compute the stepwise dot product of the filter with a local window of the spectra, with a stride of 1. We also pad the input spectra to keep the inputs and the outputs at the same size. The convolutional layer is usually followed by fully connected layers, see Figure 6.2 for an example.

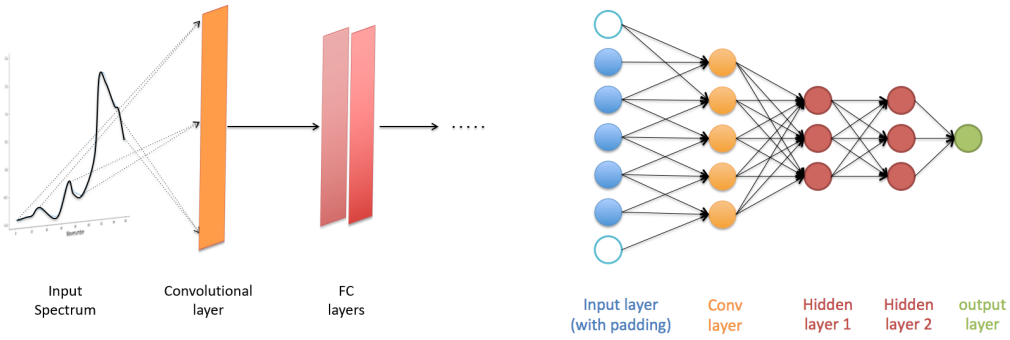


Figure 6.2: Architecture of a convolutional neural network

Considering that most of the preprocessing methods (e.g., detrend, Savitzky-Golay derivatives) in spectroscopic calibration are equivalent to moving weighted average of the input spectra, we believe that a properly trained convolutional layer can replace a huge range of the classical preprocessing treatments. Since the convolutional layer can be tuned by back-propagation, we do not need to manually choose any specific preprocessing method, but just let the optimization algorithm

find the most efficient filter.

Since the input vectors are one dimensional spectra, so the filter is also a one dimensional vector. The bandwidth of the filter is related to the resolution of the input spectra. It is possible to implement multiple paralleled convolutional channels to increase the flexibility of the model, but accordingly we would need an adequate number of training samples to tune multiple convolutional filters. Here we only consider a single channel (i.e., there is only one convolutional layer) for simplicity.

6.2.3 Activation functions

Outputs of each hidden layer, including the convolutional layer and the fully connected layer, are transformed by the activation function to allow for nonlinearity. In the following discussion, we denote the input to the activation functions as x and the output as y.

Sigmoid function computes the sigmoid of x element-wise. Specifically:

$$y = \frac{1}{1+\exp(-x)}. \tag{6.1}$$

The sigmoid function has been historically popular, due to nice biological interpretations. However, saturated neurons can cause gradient vanishing: gradient flow during back propagation is almost 0 when x is not near 0. In addition, the sigmoid function is not zero-centered, which means updates on the parameters can be very inefficient.

Tanh function computes hyperbolic tangent of x element-wise. Specifically:

$$y = \tanh(x). \tag{6.2}$$

It achieves zero-centering by squashing the output into $[-1,1]$. However, the gradient vanishing issue remains[151].

Rectified linear units(ReLU) takes element-wise rectified linear of the input, where all negative values of the input are set to zero. More specifically:

$$y = \begin{cases} x, & x \geq 0 \\ 0, & \text{otherwise} \end{cases}. \tag{6.3}$$

ReLU does not saturate. In addition, it converges faster than sigmoid and tanh[152]. However, it is still non zero-centered. The main deficiency is that the outputs of negative inputs are always 0, which may cause a specific issue called "dead ReLU". When inputs into some of the ReLU neurons are always negative, such kind of ReLU neurons will never contribute to the weight updates. This happens when a bad initialization or learning rate is assigned to the model.

Exponential linear units(ELU) is very similar to ReLU, while the negative part is exponential instead of zero. More specifically:

$$y = \begin{cases} x, & x \geq 0 \\ \exp(x) - 1, & \text{otherwise} \end{cases}. \tag{6.4}$$

It has most of the benefits of ReLU, but the neurons cannot become "dead". It is also closer to zero-centered, which makes weight updates more efficient. In addition, there is a negative saturation regime which makes it robust against some noises[153]. One minor weak point is that calculating $\exp(x)$ is slow, especially when the network is relatively large.

6.2.4 Regularization

L2 regularization is probably the most common form of regularization. It has been applied to many different machine learning schemes. By penalizing the sum of squared amplitude of the weights in the model via $\frac{1}{2}\lambda w^2$ (where λ is the regularization parameter, and w is the weight), L2 regularization encourages the model to use all the neurons, rather than a few principal neurons. The gradient of the L2 regularization term is λw, which means every weight is steadily shrunk towards zero whatever its current sign.

Dropout is a simple way to prevent neural networks from overfitting[154]. The idea can be illustrated by Figure 6.3. During the training process, we randomly activate a proportion of neurons in FC layers, and only update the parameters of the selected neurons. During testing no dropout is applied. Dropout is simple but efficient. It is widely used in modern practical neural networks, especially in large ones. Some recent studies[154][155] also illustrate its principles and relations with other regularization techniques.

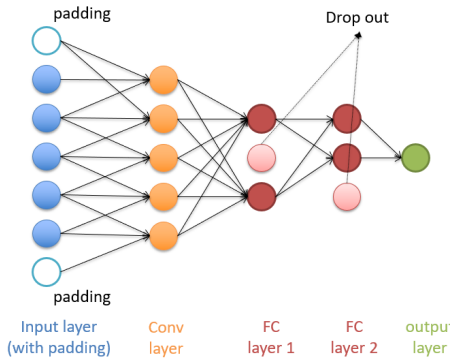


Figure 6.3: Illustration of dropout: two random neurons in the fully connected layers are dropped out during training

6.2.5 Loss functions

The loss function adopted in this research is the mean squared errors (MSE) of predictions plus L2 penalty on all weights, including weights in the convolutional layer and the fully connected layers:

$$Loss = MSE + \frac{1}{2}\lambda \sum w_i^2. \tag{6.5}$$

6.2.6 Optimization

Back-propagation is the basic optimization method used in training neural networks[156]. It calculates the error contribution of each neuron by evaluating some training samples (normally a batch of data drawn from the whole training set). It is normally used by the gradient descent optimization algorithms (including the Adam optimizer[157], which is used in this study) to tune the parameters in the model. The gradient of the loss function is calculated and then distributed back-

wards through the whole network.

Batch optimization is a necessary tool to speed up the optimization process for large datasets. When the training set is relatively small, we can simply use all the samples to tune the model simultaneously. However, when the training set is too large, it is very slow to evaluate the contribution from all the samples in each update. This is critical for neural networks because the parameters are updated during repeated forward/backward passes. Normally we only evaluate a small proportion of the training samples, which is called a mini-batch, for each update. The term 'epoch ', instead of iteration (or loops), is used to indicate the training process. One epoch means one forward/backward pass of all the training samples (i.e. 1 $epoch = \frac{train\ set\ size}{batch\ size}\ iterations$).

Adam optimization is a recently proposed optimizer for neural networks[157], and is widely considered the best optimizer currently available. It has many beautiful features, such as rescaling invariance, suitability for non-stationary loss function, and automatic learning rate annealing. When using the Adam optimizer, we need to predetermine 4 hyperparameters: β_1 (decay parameter for the gradient, normally set to 0.9); β_2 (decay parameter for the squared gradient, normally set to 0.999); α (learning rate) and ε (a very small number to prevent division by 0, for example 10^{-6}). Adam is often recommended as the default optimizer. There is also a nice research benchmarking other popular stochastic optimizers[158] as optional choices.

6.3 Results and discussions

6.3.1 Experiment 1: Self-preprocessing, learning rate, regularization and dropout

This part of study is based on dataset 2, protein calibration on wheat flour. The CNN model is constructed as shown in Table 6.1:

CNN structure for dataset 2			
Layer name	Input dimension	No. of parameters	Output dimension
Input	161		161
Conv layer	161	$5 \times 1 \times 1$	161
FC 1	161	$161 \times 36 + 36$	36
FC 2	36	$36 \times 18 + 18$	18
FC 3	18	$18 \times 12 + 12$	12
Output	12	$12 \times 1 + 1$	1

Table 6.1: CNN structure for dataset 2

Initial inputs to the CNN model are the normalized (in column axis) spectra. Notice here we do not use any other preprocessing methods, while in PLS regression spectra were preprocessed by the first derivative followed by SNV. The preprocessing methods for the PLS regression model were selected by cross-validation. There are some predetermined hyperparameters in the CNN model: learning rate = 0.01, L2 regularization parameter $\lambda = 0.005$, batch size =256, no dropout. Later we will discuss the impact of these hyper-parameters in a greater detail.

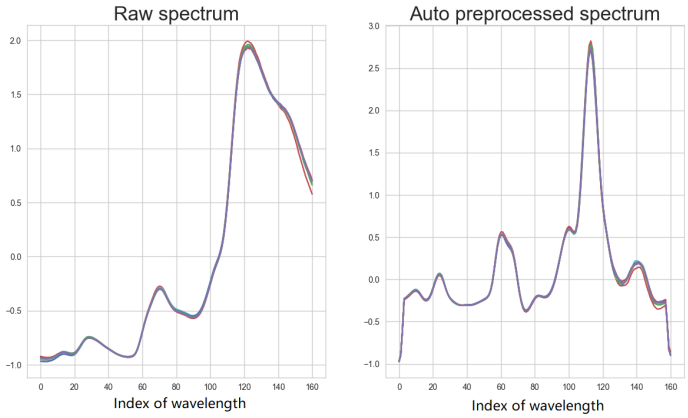
The CNN model is compared to the PLSR model (PLS factors: 8, preprocessing: 2 side points, 2^{nd} order polynomial fitted 1^{st} derivative followed by SNV). The PLSR was tuned by cross-validation. The results are showed in Table 6.2. Since the 95% confidence interval on the ratio of SEP does not include 1, we can conclude that the prediction precision of the CNN model is significantly better then the PLSR model on the investigated test set.

Model	RMSEC	RMSEP	SEP
PLSR	0.329	0.291	0.290
CNN	0.254	0.230	0.229
CI			(1.229,1.310)

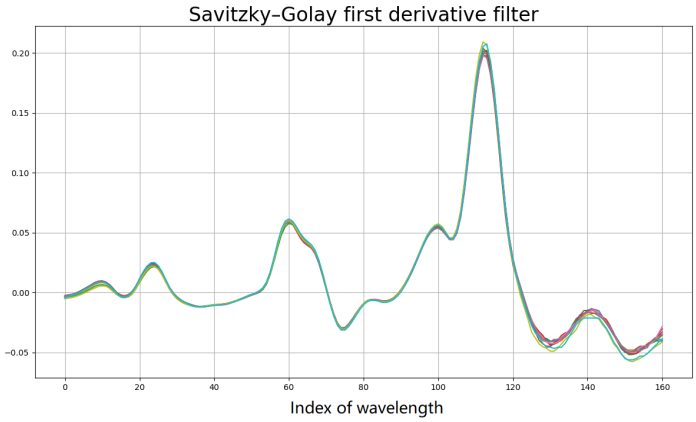
Table 6.2: Comparison of the results of the two models on dataset 1, CI stands for 95% confidence interval of the ratio on SEP (PLSR/CNN). Since the interval does not include 1, the CNN model is significantly better than the PLSR model on the test set.

6.3.1.1 Automatic preprocessing

As introduced in section 6.2.2, the convolutional layer can transform the spectra to optimize performance of the subsequent regression scheme. Since the convolutional layer was trained automatically by the optimizer, we visualize the output of the convolutional layer to study the actual transformation. The result is shown in Figure 6.4.



(a) Spectroscopic transformation by the convolutional layer



(b) SavitzkyGolay first derivative filter:bandwidth=5, polynomial fitting order=2

Figure 6.4: (a) 20 random spectra in dataset 2 and the corresponding outputs of the convolutional layer. (b) 2 side points, second polynomial fitting Savitzky-Golay first derivative on the same 20 spectra.

Based on the plots, we can see that the transformation of the convolutional layer is similar to a Savitzky-Golay first derivative filter. However, they are not exactly the same. Outputs of the convolutional layer correspond to a skewed and

rescaled standard first derivative filter. The convolutional layer continually tunes the parameters in the filter, until it finds the best form of preprocessing. This means the spectroscopic preprocessing done by the convolutional layer is more flexible. More importantly, the traditional way of choosing the preprocessing methods is via trial-and-error experiment, which is quite labour and time consuming. The proposed CNN method saves a lot of effort when building new calibrations.

6.3.1.2 Regression coefficients

We used the method introduced in section 2.3.3 to plot the regression coefficients of the CNN model. The results are shown in Figure 6.5. We randomly drew 100 spectra from the training set, and calculated the corresponding regression coefficient curves. As we can see in Figure 6.5, the two models have similar profiles. The CNN model recognizes a few different signals in the spectra, but overall it is very smooth and robust.

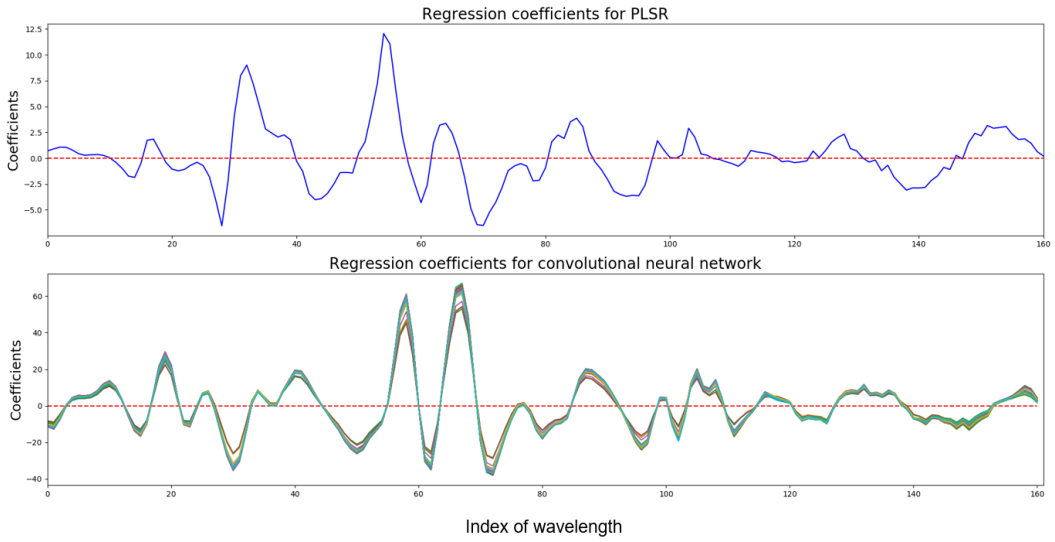


Figure 6.5: Comparison of regression coefficients trained by PLSR and CNN

When training a CNN model, we need to determine a few hyperparameters: λ (regularization parameter), learning rate, activation function and dropout ratio. In what follows, we explore the impacts of these hyperparameters on the CNN

model, and hence provide insights how to determine them. It is worth noting the best practice to tune the hyperparameters is by coarse-fine grid searching on a separate validation set. However for NIR calibration, a separate validation set is not always available, so cross-validation can be used instead. We propose some common choices for these hyperparameters. They can be used as the default values when lacking resources to tune, or as indication of where to perform a grid search.

6.3.1.3 Learning rate

During the training process, we randomly drew a batch of samples from the training set, and then updated all the parameters based on the gradients. Learning rate determines the step of each update. If the learning rate is too large, the update becomes too aggressive, which may lead to a bad local optimum. If the learning rate is too small, the loss function converges very slowly.

Krizhevsky proposed a way to determine the learning rate[159]:

$$learning\ rate = 0.01 \times batch\ size/256 \tag{6.6}$$

For example, if we fix the batch size at 256, then the learning rate should be 0.01. We found the heuristic proposed is valid on our tested datasets. Refer to Figure 6.6 for the results. We plot three training loss curves through the optimization process, with different learning rates. When we set the learning rate to 0.01 (red line), the loss decays consistently and efficiently. We think it is a good learning rate. When we set the learning rate as 0.05 (blue line), which is too high, the objective function finally converges to a bad local optimum. When learning rate is too small (green line, learning rate =0.005), we found it slow to converge (but will eventually converge to a good local optimum). In summary we recommend using 0.01 as default learning rate when the batch size is 256.

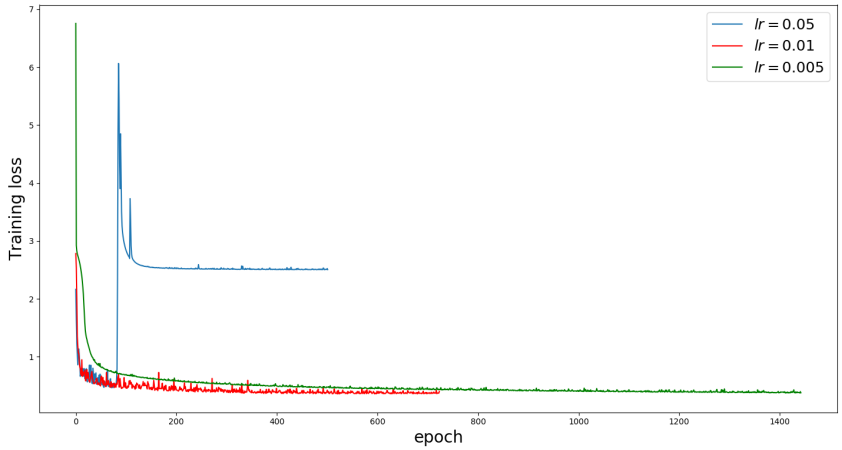


Figure 6.6: Optimization process with different learning rate

6.3.1.4 Activation function

Referring to Table 6.1, we inserted an activation function after each hidden layer. The full structure is: Input - Conv layer - activation - FC1 - activation - FC2 - activation - FC3 - activation - output. We tested 4 different activation functions: Sigmoid, ReLU, ELU, ELU + ReLU. In ELU+ReLU we used ELU after the convolutional layer but ReLU after the FC layers. The final RMSEP on the test set are showed in Figure 6.7. We show PLSR as the benchmark method as well.

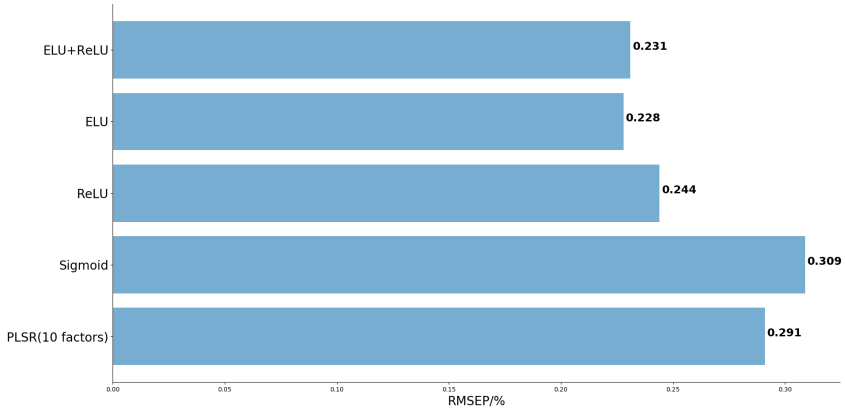


Figure 6.7: Performance of different activation functions

1. We found that the sigmoid function is not suitable in our CNN model. This is due to the fundamental limitations of the sigmoid function (gradient vanishing, non-zero-centered). The result is worse than the PLSR method.

2. ELU is slightly better than ReLU. The ELU+ReLU configuration is very close to using ELU solely, we consider they have equivalent performance.

3. The biggest advantage of ReLU over ELU is low in computational cost and fast convergence. In ReLU all the negative outputs are set to zeros while positive outputs are unchanged. In contrast ELU involves calculating exponentials, which is one of the most expensive operations. Refer to Figure 6.8, the ReLU CNN converges in around 600 epochs while ELU takes roughly 1000. A good trade-off is using ELU after the convolutional layer, to keep the desired nonlinearity, then ReLU after the FC layers to improve efficiency in training. This configuration benefits from both of training speed and flexibility. It is worth noting that we found the training times with different activation functions vary from 70 seconds to 130 seconds (on our computational system, a 2.50GHz quad core CPU), which might be trivial in many practical cases.

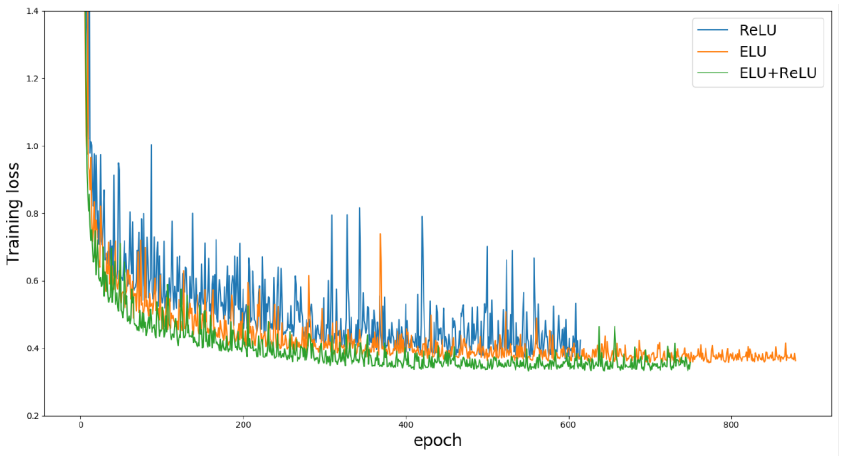


Figure 6.8: Convergence speed of 3 different activation functions

6.3.1.5 Regularization parameter

L2 regularization is the most efficient way to prevent CNN models from overfitting. We performed an experiment to learn the impact of the regularization parameter (λ) on the final CNN model. We trained the CNN models with different amounts of L2 regularization. Here we also plot the corresponding regression coefficients, refer to

Figure 6.9.

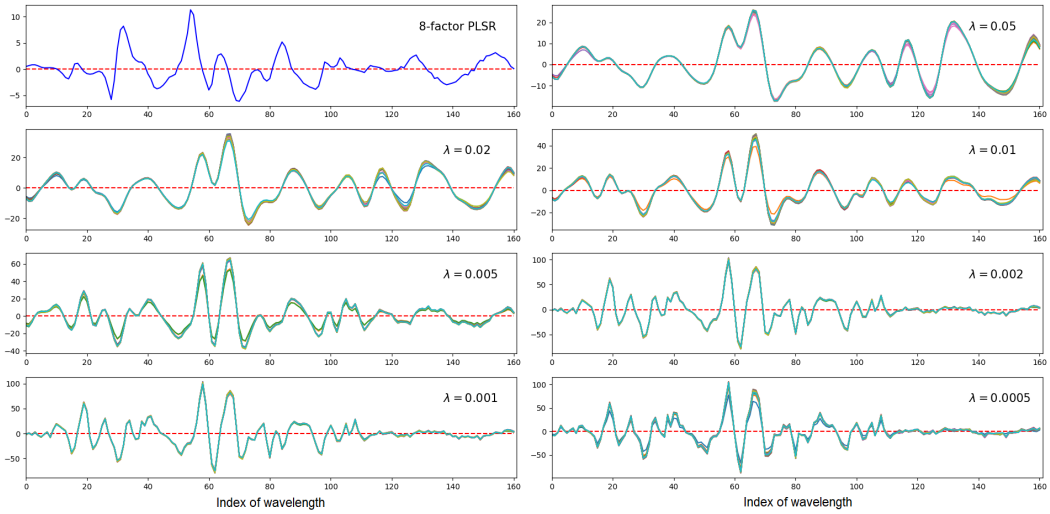


Figure 6.9: Regression coefficients trained by different regularization parameter. Plot on the top left (blue curve) is a 8-factor PLS regression model.

We found that λ controls the noise level of the regression coefficients. In PLSR, we use the number of factors to tune the complexity (hence the noise level) of a prediction model. λ has a similar effect but does not change the overall profile of the regression coefficients. When we apply L2 regularization to a CNN model, we always use the same principal signal and change tolerance on presence of small signals, which can be noise.

When choosing λ, we should refer to the cross-validation accuracy, i.e. the root mean squared error of cross-validation (RMSECV). However, for NIR calibration, we do not always choose the model with smallest RMSECV, which can be overfitting. Instead we often prefer the most robust model with an acceptable performance. In Figure 6.10 we plot the Fourier transform on regression coefficients of different models. For the CNN models, magnitudes were averaged out over 100 coefficient curves on the training set. We can clearly see how the noise level grows when the PLSR and the CNN models become more complex. In Table 6.3 we show the RMSECV, RMSEP, SEP and noise level when changing number of PLS factors and λ.

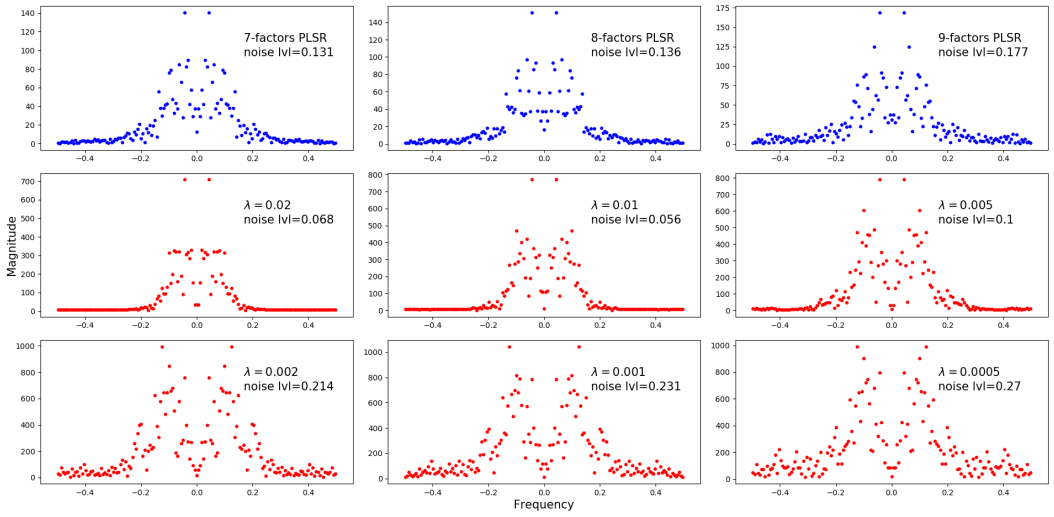


Figure 6.10: Fourier transform on regression coefficients of different PLSR (blue) and CNN models(red). Noise level was calculated by the ratio of magnitude of high frequency (≥ 0.3) components.

λ/PLS factors	RMSECV	RMSEP	SEP	Noise level
7-factor PLSR	0.354	0.321	0.320	0.131
8-factor PLSR	0.329	0.291	0.290	0.136
9-factor PLSR	0.310	0.286	0.286	0.177
0.05	0.419	0.365	0.364	0.045
0.02	0.371	0.325	0.325	0.068
0.01	0.315	0.285	0.284	0.056
0.005	0.254	0.230	0.229	0.100
0.002	0.233	0.223	0.222	0.214
0.001	0.214	0.222	0.222	0.231
0.0005	0.205	0.215	0.215	0.270

Table 6.3: λ selection in cross validation. Two candidate CNN models are shaded with blue and yellow color

Some discussions on the results:

1. We choose the model with $\lambda = 0.005$. Further decreasing regularization on the CNN model does not significantly improve RMSECV, but doubles the noise level.

2. Compared to a 8-factor PLSR model, we can see that the $\lambda = 0.005$ CNN

model has a slightly lower noise level, but a significantly better performance. It is also worth noting the $\lambda = 0.01$ CNN model has similar RMSEP and SEP with the 8-factor PLSR model, but a much smaller noise level on the regression coefficients. The same conclusion is obtained from Figure 6.9, the CNN model with $\lambda = 0.01$ is very smooth and clean.

3. The amount of regularization that should be added to a CNN model is dependent on the spectral quality, the target property and the structure of the neural network. When we use the same spectroscopic system and CNN architecture, the regularization parameter should roughly scale with the square of the target property concentration. Refer to appendix B for some examples of our experiments. This can help us assign initial search range for λ.

6.3.1.6 Dropout ratio

Dropout is another way to add regularization to a CNN model. Normally dropout is only employed after FC layers. In this test we added one additional dropout layer after FC1, FC2 and FC3. By tuning the dropout rate we found that the regression coefficients become less noisy, at the price of losing fitting capability. Refer to Figure 6.11 and Table 6.4.

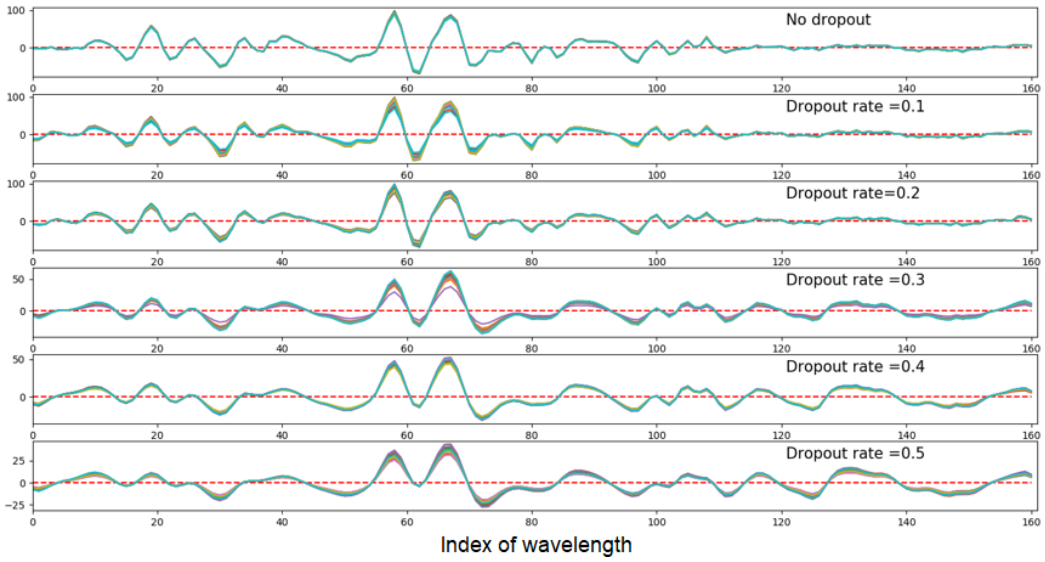


Figure 6.11: Regression coefficients trained with different dropout rate

Dropout rate	RMSECV	RMSEP
0	0.254	0.230
0.1	0.277	0.231
0.2	0.291	0.248
0.3	0.336	0.315
0.4	0.359	0.289
0.5	0.383	0.349

Table 6.4: Effect of dropout in a CNN calibration

From figure 6.11 we can observe:

1. When no dropout applied, there is more random noise in the regression coefficients. Especially on the right hand side of the spectrum, there is only random noise near zero.

2. the higher dropout rate, the less noise in the model. Particularly, when dropout rate = 0.5, there is no apparent Gaussian noise in the regression coefficients.

3. Another key impact of dropout rate is nonlinearity of the model, which can be interpreted by the difference of the curves in the same model. For example, when no dropout applied, there is no obvious difference between curves, which means the whole model is very close to a linear model; When dropout rate = 0.5, difference between the curves is more visible, which means the nonlinearity of the model is higher.

From the results we can conclude that the impact of dropout is very gentle, especially when dropout rate is small. By increasing the dropout rate the cross-validation error continuously goes up, due to reduced usable features in the network, but the out-of-sample prediction accuracy is rather consistent.

Based on this dataset and our proposed CNN structure we find that the benefit of dropout is not very obvious. In general we can just set the dropout rate to 0. When

a robust and smooth calibration model is desired, we recommend using dropout as an additional regularization tool, in addition to L2 regularization.

6.3.2 Experiment 2: Application to large datasets and scalability

Here we show another example of using the CNN method for NIR calibration. The experiment was done on dataset 3, which is ash on wheat flour. The CNN model was trained on a very large training set (6987 samples), then evaluated on a separate test set. The samples in the test set were measured under the same spectroscopic system. However, training samples and test samples have different varieties, origins, physical conditions, etc.From Table (6.5), we see for both the PLSR and the CNN calibration models, the training and the cross-validation errors are much smaller than the test error, which indicates the structures of the two datasets are different. However, we observe that a properly regularized CNN model is more robust and more accurate on the test set.

For the PLSR calibration, we applied the second order polynomial detrend + SNV to the raw spectra. Cross validation on the training set suggests using 10 PLS factors. RMSEP and SEP on the separate test set are 0.094% and 0.088% correspondingly.

We used the same CNN structure as introduced in experiment 1, refer to Table 6.1 for detailed description. Raw spectra were normalized in column axis before feeding to the CNN model. ELU activation function was employed. During the training process, we fixed the batch size at 256, and learning rate at 0.01 (corresponding to Equation 6.8). Similar to model selection in PLSR, we used cross-validation to choose regularization parameter. No dropout was applied. A summary of the cross validation is shown in Table (6.5).

λ	RMSECV	RMSEP	SEP	Noise level
		... ...		
6×10^{-4}	0.056	0.093	0.081	0.028
4×10^{-4}	0.053	0.089	0.077	0.040
2×10^{-4}	0.051	0.086	0.076	0.015
1×10^{-4}	0.048	0.085	0.075	0.036
8×10^{-5}	0.045	0.091	0.077	0.081
6×10^{-5}	0.058	0.083	0.074	0.122
		... ...		

Table 6.5: CNN model selection on dataset 3: fine tuning on λ.

In the following discussion, we set the regularization parameter at 1×10^{-4}, since it has the best RMSECV/noise level trade-off. First we carried out a hypothesis test to compare the prediction accuracy between the PLSR model (with 10 factors) and the CNN model. Results showed that 95% confidence interval of the true ratio of the SEP is: $PLSR/CNN = (1.198, 1.325)$. Since this interval does not include 1, we can conclude that the prediction error of the CNN model is significantly smaller than the PLSR model. Refer to Table 6.6 for a brief summary:

Model	RMSECV	RMSEP	SEP	Noise level
PLSR	0.052	0.094	0.088	0.165
CNN	0.048	0.085	0.075	0.036
CI			(1.178, 1.314)	

Table 6.6: Comparison of the two calibration methods on dataset 3. PLSR: 10 factors, 2^{nd} order polynomial detrend +SNV; CNN: $\lambda = 10^{-4}$, learning rate=0.01, no dropout. CI stands for the 95% confidence interval for the ratio of SEP.

In Figure 6.12, we compare the prediction - reference plots and the regression coefficients of the two models. We can clearly observe that while the CNN model is more precise than the PLSR model, the regression coefficients of the CNN is also smoother than that of the PLSR model. Specifically, the noise level of the CNN model (0.036) is much smaller than that of the PLSR model (0.165). This indicates that the enhanced performance does not come from overfitting on the dataset we used.

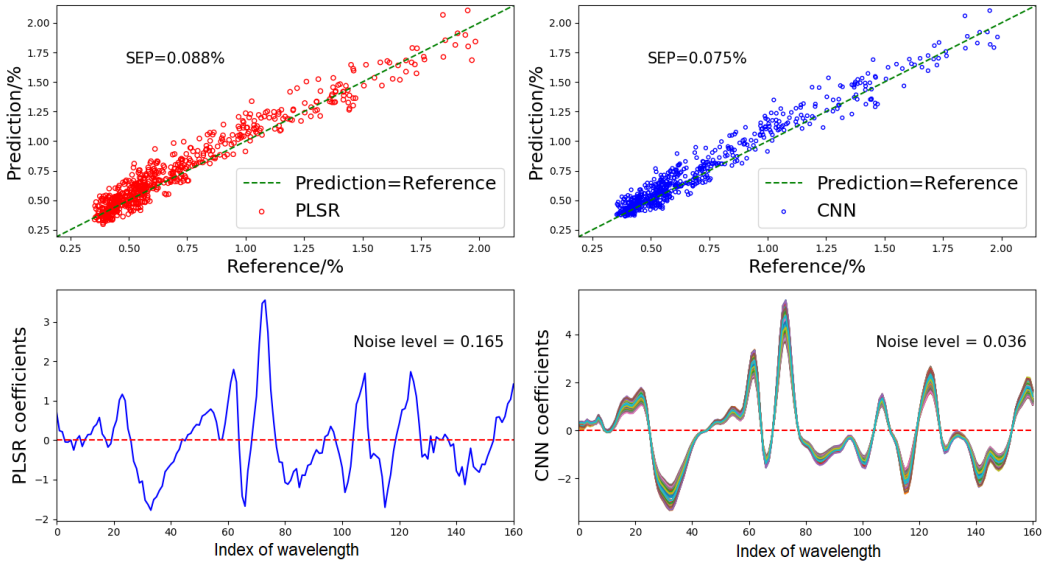


Figure 6.12: Comparison of the regression coefficients between the PLSR model and the CNN model. PLSR : 10 factors, detrend2 + SNV applied to the raw spectra. CNN model: $\lambda = 0.0001$, configuration refer to Table 1

We also investigated the automatic preprocessing done by the CNN model, refer to Figure 6.13[1]. Different from experiment 1, the automatic preprocessing on dataset 3 is not based on the first derivative. It appears that the outputs of the convolutional layer[2] are skewed and rescaled raw spectra.

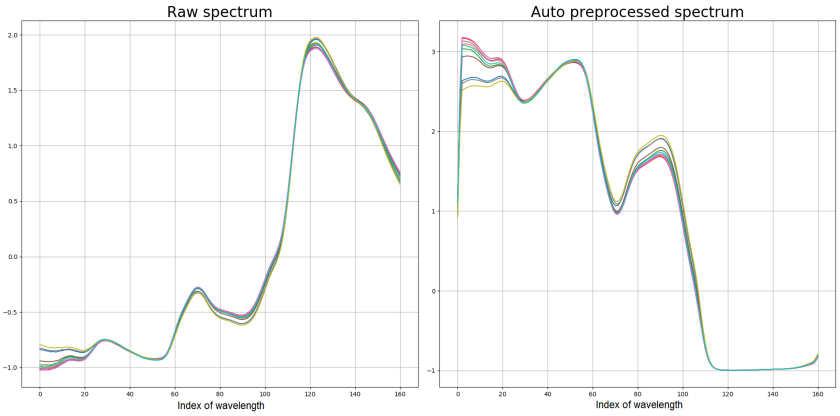


Figure 6.13: Automatic preprocessing by the convolutional layer. We randomly draw 10 spectra from the training set (on the left) and plot the outputs of the convolutional layer (on the right)

.

[1]Raw spectra were processed by SNV purely for visualization purpose.

[2]when we say "the outputs of the convolutional layer", we mean the outputs of the activation function attached to the convolutional layer.

6.3.3 Experiment 3: Application to small datasets

Our proposed method can be applied to small datasets. However, the risk is that when the number of training samples is not adequate, it is very likely we are not able to get the best model. Here we present an example on dataset 5.

Dataset 5 is on protein content in whole wheat grain. There are 415 samples in the training set and 108 samples in the test set. We used the same CNN architecture as introduced in experiment 1 and 2. For dataset 5, we have no information about the instrument and the commodity features. In practice, some neural network structures should be adjusted accordingly (for example, filter width in the convolutional layer). For simplicity, here we just use the same CNN structure as before.

In the PLSR model, we use the second order polynomial detrend + SNV as the preprocessing methods, according to the CV result. CV also suggests using 8 PLS factors. The final prediction accuracy on the test set is: RMSEP= 0.425%; SEP=0.411%.

In the CNN model, we recommend that users adopt the second-order optimizers for small datasets. For example, in this experiment we used the Limited-memory BFGS optimizer, which works very well in full batch, deterministic mode (i.e., we do not draw a batch of samples from the training set in each update, but use the whole dataset every time). The benefit of such kind of optimizers is a much faster convergence. We set the regularization at 0.003 as suggested by the CV. The final prediction accuracy on the test set is: RMSEP=0.420; SEP=0.414. A summary of the results is shown in Table 6.7.

Model	RMSECV	RMSEP	SEP	Noise level
PLSR	0.439	0.425	0.411	0.278
CNN	0.425	0.420	0.414	0.080
CI			(0.981,1.103)	

Table 6.7: Comparison of the two calibration methods on dataset 5. CI stands for the 95% confidence interval for the ratio of SEP.

The regression coefficients are shown in Figure 6.14. It is obvious that while the prediction performance of the CNN model and the PLSR model are quite close, the noise level in the CNN model is much smaller than that of the PLSR model. In addition, from Figure 6.14 we can observe that the nonlinearity is more apparent than in dataset 1 or dataset 3, namely the variance of the regression coefficient curves from the CNN model is bigger. This is due to the diversity of the whole grains (surface condition, hardness, color, etc.). For flour, the samples were thoroughly homogenized before being measured, so the nonlinearity only comes from the NIR - target constituent correlation. Obviously for whole grains physical conditions of the grains also contribute to a huge part of nonlinearity in the calibration model.

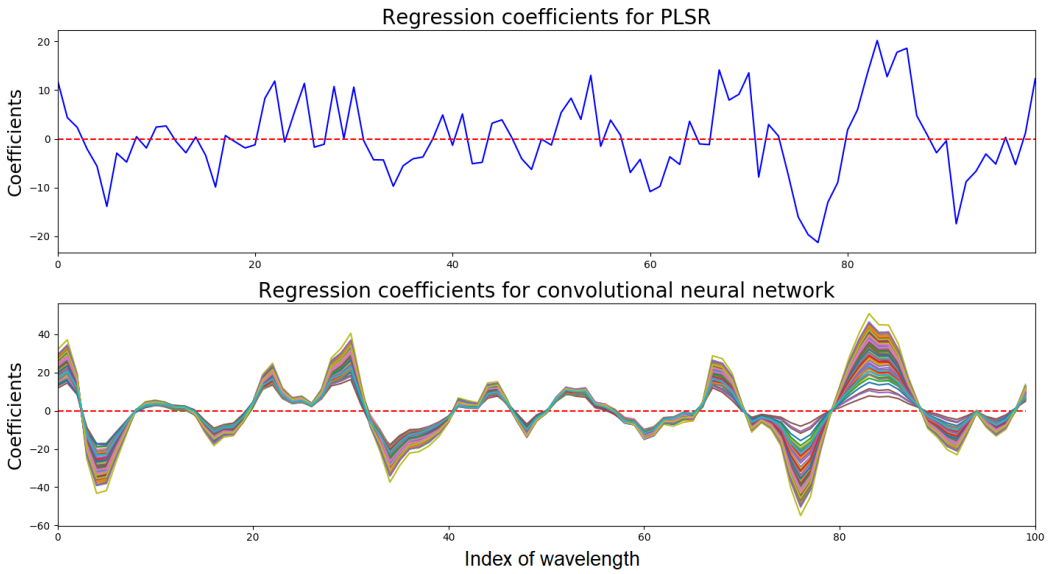


Figure 6.14: Comparison of the regression coefficients of the two calibration methods on dataset 5

Another issue we found is that ReLU is not suitable for the small dataset. When using RelU as the activation function, we can easily get a bad local optimum solution. Refer to Figure 6.15 for an example. The calibration model is truncated for part of the samples in the test set. We think it is related to the sparsity of the ReLU function. When the inputs to the ReLU function are negative, the outputs are always zero, which means such neurons are ”dead”. If all the neurons in any of

the layers are all ”dead” then the whole network is not active anymore. For small dataset there is a certain risk that the weights are not properly tuned, especially for some extreme inputs.

As discussed in the previous section, the major advantage of ReLU over ELU is low computational cost. For a small dataset, when we use the second order full batch optimizer, convergence speed is not a issue anymore so we can always use ELU to avoid such bad local optima.

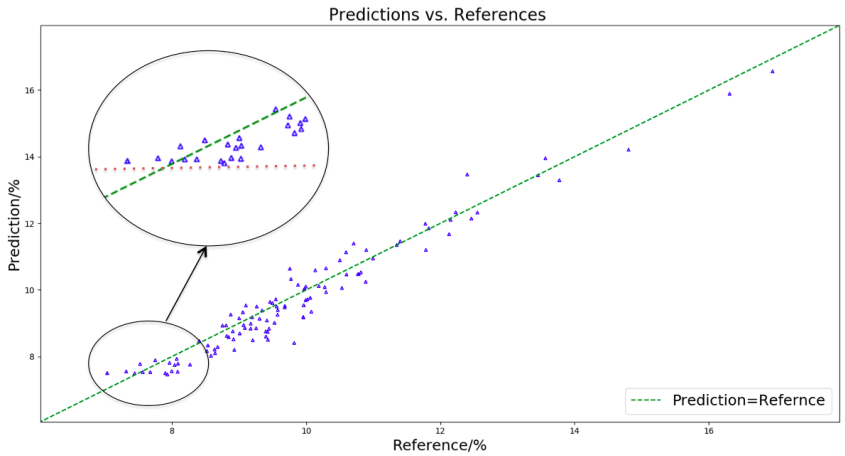


Figure 6.15: When using ReLU as the activation function, there is a ”dead” region for samples low in constituent concentration. Prediction values are truncated.

6.4 Conclusions

We have discussed how to use CNN for NIR calibration. We have done systematic tests on three different NIR datasets, to show the advantages of the CNN method over the PLSR method. A summary of our key results:

1. The convolutional layer can play a role of spectral preprocessor. This kind of layer can be automatically trained, which saves effort in selecting preprocessing methods.

2. By visualizing the regression coefficients and calculating the high frequency components in the Fourier domain, we compared the introduced CNN method with

the PLSR method. We have shown that the CNN models are not only more accurate than the PLSR models (smaller RMSEP and SEP), but also less noisy and more robust

3. We have systematically compared different choices of hyperparameters in the CNN models, including activation function, learning rate, regularization parameter, and dropout rate. All the hyperparameters in the model can be selected by cross validation. We also recommended some default initial configurations for common applications.

4. We have tested the proposed method on 3 different NIR datasets of different sizes. We have illustrated the CNN method can be used for general NIR calibrations, even when the dataset size is not that large.

Chapter 7

General Conclusions

In this thesis, we have introduced three different nonlinear multivariate regression schemes for NIR calibration: kernel methods, hierarchical mixture of linear regressions and convolutional neural networks. We have compared our nonlinear methods with the benchmark regression method of PLSR on 5 real world NIR datasets. The investigated nonlinear methods all have appropriate regularization treatments: kernel methods use the length scales to avoid overfitting; HMLR as introduced in our study uses PLS dimension reduction and a prior distribution on weights for regularization; in CNN we use L2 regularization to penalize parameters. Results showed our proposed nonlinear regression methods had improved predictive capability and many other benefits.

The investigated kernel methods including LS-SVM and GPR are nonparametric. The two methods have very similar formulations and predictive capability. Both of them use the whole training set as the calibration library, and predictions on new observations are based on their spectral covariance with the training samples. Kernel methods are parameter free. Hyperparameters including signal level, noise level and length scale are used to balance the fitting capability and generalization performance of the model. In LS-SVM we used LOO-CV to tune the hyperparameters while in GPR gradient-based evidence maximization were applied. Both of them exhibited improved predictive performance on our NIR dataset when compared to the PLSR method. We noticed the biggest issue is that

for non-parametric methods, the computational cost of prediction scales with the size of the training set. For example, if we are only interested in point prediction, then we need to evaluate $E(\hat{f}_{I+1}|\mathbf{X},\mathbf{y},\mathbf{x}_{I+1}) = \mathcal{K}_{GP}(\mathbf{x}_{I+1},\mathbf{X})[\mathcal{K}_{GP}(\mathbf{X},\mathbf{X}) + \boldsymbol{I}/\gamma]^{-1}\mathbf{y}$. The last two terms $[\mathcal{K}_{GP}(\mathbf{X},\mathbf{X}) + \boldsymbol{I}/\gamma]^{-1}\mathbf{y}$ can be pre-calculated and stored in memory (if storage of a huge model is not a problem). Making a prediction on the new observation involves N computations of exponential squared distance. This can be non-trivial, for example on our dataset 3, which has a training set size of 6987.

In Chapter 4, we used RBF kernel in both LS-SVM and GPR, which means the distance matrix vary isotropically along all directions of the input spectrum. RBF kernel has been selected arbitrarily to help us quickly understand some fundamental characteristics of the kernel methods. In practical, RBF is very unlikely the most suitable distance measurement for all the spectroscopic calibrations. One interesting future work here is implementing metric learning in our calibration process. Metric learning was originally proposed for image clustering, with the target of maximizing the inter-class variatoins and minimizing the intra-class variations[160]. This can be done by optimizing the similarity, i.e. the distance between two input vectors (in our case, spectra) defined by:

$$d_M(\mathbf{x}_i,\mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T\mathbf{M}(\mathbf{x}_i - \mathbf{x}_j)}, \tag{7.1}$$

where $\mathbf{M}$ is a semi-definite matrix, which can be decomposed as:

$$\mathbf{M} = \mathbf{W}^T\mathbf{W}. \tag{7.2}$$

Suppose that vector $\mathbf{x}$ has the length of d, then in Chapter 4, where we used the RBF kernel, $\mathbf{W} = \mathbb{1}^{d\times d}$. In another extreme case, where M is a diagonal matrix ($\mathbf{W} \in \mathbb{R}^{d\times d}$), the length scale on any direction of $\mathbf{x}$ can be different. This is called automatic relevance determination (ARD) in Gaussian process. Length scales on $\mathbf{x}$ can be tuned by the importance, and hence can determine their impact when making the prediction.

In a more general case, $\mathbf{W} \in \mathbb{R}^{p \times d}$, where $p \leq d$. As a result, equation (7.1) can be modified as:

$$d_M(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{W}^T \mathbf{W} (\mathbf{x}_i - \mathbf{x}_j)} = \| \mathbf{W}\mathbf{x}_i - \mathbf{W}\mathbf{x}_j \|_2, \tag{7.3}$$

which is also called Mahalanobis metric learning, is equivalent to finding a linear mapping that transforms every single sample $\mathbf{x}_i$ into a lower dimensional space ($\mathbb{R}^{p \times 1}$), where Euclidean distance is calculated. The mapping function $\mathbf{W}$ can be learning by optimizing some loss functions, for example squared errors or log marginal likelihood.

The second nonlinear regression method, HMLR is as simple as PLSR. It improves the flexibility of the model simply by mixing multiple PLSR models. In this research we proved that by using just 2 PLSR component models the consistency and performance of prediction could be improved. The whole training process including data segmentation is fully automated by using the EM and variational inference algorithm. Extra computational cost introduced by the gating function and additional linear regressions is ignorable compared to the kernel methods. Since the model is built on top of PLSR, we believe it is as robust as the PLSR method. In addition we can still use the leverage and spectral residual as indicators for outliers, just like PLSR. The only issue is that there is no mature package for such kinds of algorithms. Users who are interested in the method need to develop the whole calibration framework from scratch, which requires a lot of knowledge of graphical models and can be quite time-consuming. In addition, it is not very clear how to build proper deep nested hierarchical models. Immature optimization of the codes may also lead to a reduced performance.

One biggest limitation in our proposed structure is that the original spectrum were first processed by PLS or PCA for dimension reduction, and then sent to the hierarchical linear modelling. As a result, dimension reduction is separate from our

learning process. In addition, all the component models have the same number of PCs, which is not necessarily the most suitable solution. One promising future work is to make the dimension reduction a probabilistic process, and separate for each component model. The probabilistic dimension reduction function can be trained together with other parameters in the hierarchical linear modelling, by optimizing the overall likelihood. This increase flexibility of the model, and some component model will use less components, which means more robust when making predictions.

CNN has become very popular in many machine learning applications. We have proved that the CNN method can be a one of the best nonlinear calibration methods on NIR applications. Compared to the other two proposed methods, CNN has lots of advantages. It is parametric, the size of the model is fixed. For a ”shallow” CNN model we do not need to worry about the computational cost. There are plenty available packages such as tensorflow, caffe, torch and theano which help the users to build complex CNN models. In addition, with the convolutional layer we do not need to manually select spectral preprocessing methods, which can be advantageous in some cases. However, the users need to explore the best structure of the CNN model for use with the dataset of interest. Some parameters, e.g. the filter width of the convolutional layer, are related to the physical characteristics of the system. We need to pay special attention on the choices of these parameters in practical applications.

In Chapter 6 we have seen how L2 regularization and dropout can be used to regularize the CNN model. In particular we see that dropout adds turbulence to the features that are used in prediction, which can improve robustness of the model. Another similar approach in spectroscopic analysis is to add turbulence to the raw spectra, then new spectra with noises (for example, Gaussian white noise) can be created. In such way a lot of artificial data points can be generated to expand the training set. The concept is very similar to ”data augmentation” in deep

learning, which helps to increase the size of the dataset. However, one noticeable issue is that noise added to raw the spectra should be realistic in the practical system. If added noise is less likely to be observed in the spectroscopic system, then such kind of data augmentation will lead the training process to the wrong direction.

In Chapter 6 we have demonstrated how CNN method can generate new pretreatment methods, and how CNN models outperform PLSR models. However, we have not validated which part has bigger contribution to the predictive capability, i.e. whether the new pretreatment methods have bigger impact on the final prediction accuracy, or the nonlinearity introduced by the FC layers are more critical. One interesting future work will be experiment on the decomposition of the CNN model. Comparison of a convolutional layer + PLS regression and the original CNN structure can help us with understanding whether we can further simplify the CNN structure by using PLS regression instead of stacked FC layers.

We have proved how the CNN method can be used in spectroscopic analysis, inspired by a wide range of researches in computer vision discipline. Another popular research area in deep learning is on nature language processing (NLP) and time series analysis, where recurrent neural network (RNN) plays a big role. RNN can deal with dependence of the input sequence, for example a sentence or a time series sequence. Recent works including long short-term memory (LSTM)[161] and gated recurrent network (GRU)[162] solves training issues with the original RNN methods (gradient vanishing and gradient exploding) and improve the performance of the method. Recent researches try to combine CNN with RNN for image processing[163], scene labelling[164] and music classification[165]. One meaningful future work for us is to explore whether we can use the same idea for spectroscopic calibration. We can use the convolutional layer as the feature extractor and automated preprocessor, and the following recurrent units are used to learn contextual dependencies. Since the inner dependency of spectrum can be handled by the recurrent units (GRU for example), we believe it has the potential

to further improve predicative capability. Such kind of hybrid model can be trained by back-propagation as well.

In summary, for those who are more used to the PLSR method, we recommend using HMLR as a conservative improvement over PLSR. In HMLR we can still interpret the model obtained in the traditional way; when the training set has a medium size, for example from 500 to 5000 samples, we suggest considering kernel methods as an option; we always encourage exploring suitable CNN structures for any types of dataset, and it is a good practice to visualize the CNN models obtained to avoid over-fitted ”black-box” models.

Appendix A

Variational distributions for hierarchical mixture of linear regressions

Using Equation (5.17) and Equation (5.19), we can put down the expectation of all latent variables and parameter after each update.

Z:

$$
\begin{aligned}
\log q^*_{\boldsymbol{Z}}(\boldsymbol{Z}) =& \langle \log\{\prod_n p(y_n|\boldsymbol{\phi}_n, \boldsymbol{z}_n, \boldsymbol{W}, \boldsymbol{\tau}) p(\boldsymbol{z}_n|\boldsymbol{V}, \boldsymbol{\phi}_n)\}\rangle + const \\
=& \langle \sum_n \log p(y_n|\boldsymbol{\phi_n}, \boldsymbol{W}, \boldsymbol{z_n}, \boldsymbol{\tau}) + \sum_n \log p(\boldsymbol{z}_n|\boldsymbol{V}, \boldsymbol{\phi}_n)\rangle + const \\
=& \langle \sum_n (\sum_{i=1}^{K+1} z_{ni}(\frac{1}{2}\log\tau_i - \frac{1}{2}\log 2\pi - \frac{1}{2}\tau_i(y_n - \boldsymbol{w}_i^T\boldsymbol{\phi}_n)^2) + \sum_{i=1}^{K} t(i,\boldsymbol{z}_n) z_{ni}\boldsymbol{v}_i^T\boldsymbol{\phi}_n)\rangle + const \\
=& \langle \sum_n (\sum_{i=1}^{K} z_{ni}(\frac{1}{2}\log\tau_i - \frac{1}{2}\log 2\pi - \frac{1}{2}\tau_i(y_n - \boldsymbol{w}_i^T\boldsymbol{\phi}_n)^2) \\
& + z_{n(K+1)}(\frac{1}{2}\log\tau_{K+1} - \frac{1}{2}\log 2\pi - \frac{1}{2}\tau_{K+1}(y_n - \boldsymbol{w}_{K+1}\boldsymbol{\phi}_n)^2) \\
& + \sum_{i=1}^{K} t(i,\boldsymbol{z}_n) z_{ni}\boldsymbol{v}_i^T\boldsymbol{\phi}_n)\rangle + const.
\end{aligned}
\tag{A.1}
$$

Where $t(i,\boldsymbol{z}_n)$ is defined by Equation (5.5), and the following relationship is satisfied:

$$t(i,\boldsymbol{z}_n) = \begin{cases} 1 & z_{ni} = 1 \\ 0 & z_{ni} = 0 \end{cases}. \tag{A.2}$$

As a result we can replace $t(i,\boldsymbol{z}_n)$ with z_{ni} and eliminate $t(i,\boldsymbol{z}_n)$. Equation (A.1) can be rewritten as:

$$\begin{aligned} \log q_Z^*(Z) =& \langle \sum_n \{ \sum_{i=1}^K z_{ni} (\frac{1}{2}\log\tau_i - \frac{1}{2}\log 2\pi - \frac{1}{2}\tau_i (y_n - \boldsymbol{w}_i^T \boldsymbol{\phi}_n)^2 + \sum_{i=1}^K z_{ni} \boldsymbol{v}_i^T \boldsymbol{\phi}_n) \\ & + z_{n(K+1)} (\frac{1}{2}\log\tau_{K+1} - \frac{1}{2}\log 2\pi - \frac{1}{2}\tau_{K+1}(y_n - \boldsymbol{w}_{K+1}\boldsymbol{\phi}_n)^2) \} \rangle + const \\ =& \langle \sum_n \{ \sum_{i=1}^K z_{ni} (\frac{1}{2}\log\tau_i - \frac{1}{2}\tau_i (y_n - \boldsymbol{w}_i^T \boldsymbol{\phi}_n)^2 + \boldsymbol{v}_i^T \boldsymbol{\phi}_n) \\ & + z_{n(K+1)} (\frac{1}{2}\log\tau_{K+1} - \frac{1}{2}\tau_{K+1}(y_n - \boldsymbol{w}_{K+1}\boldsymbol{\phi}_n)^2) \} \rangle + const, \end{aligned} \tag{A.3}$$

where we have:

$$z_{n(K+1)} = 1 - \sum_{i=1}^K z_{ni}. \tag{A.4}$$

Eliminate all terms irrelevant with z_{ni}, further we have:

$$\begin{aligned} \log q^*_{z_{i\neq K+1}}(z_{i\neq K+1}) =& \langle \sum_n \sum_{i=1}^K z_{ni} (\frac{1}{2}\log\tau_i - \frac{1}{2}\log\tau_{K+1} - \frac{1}{2}\tau_i (y_n - \boldsymbol{w}_i^T \boldsymbol{\phi}_n)^2 \\ & + \frac{1}{2}\tau_{K+1}(y_n - \boldsymbol{w}_{K+1}^T \boldsymbol{\phi}_n)^2 + t(i,\boldsymbol{z}_n)\boldsymbol{v}_i^T \boldsymbol{\phi}_n) \rangle + const. \end{aligned} \tag{A.5}$$

It is quite difficult to write out the distribution of $\boldsymbol{z}_n$, but it is relatively easier to find out a set of binary varaibles $\boldsymbol{z}_n$ which can maximize $\log q_Z(Z)$. Set $z_{ni} = 1$ if and only if i maximize the target function:

$$\begin{aligned} g(i) =& \underset{i}{argmax} \{ \frac{1}{2}\log\tau_i - \frac{1}{2}\log\tau_{K+1} - \frac{1}{2}\tau_i (y_n - \boldsymbol{w}_i^T \boldsymbol{\phi}_n)^2 \\ & + \frac{1}{2}\tau_{K+1}(y_n - \boldsymbol{w}_{K+1}^T \boldsymbol{\phi}_n)^2 + t(i,\boldsymbol{z}_n)\boldsymbol{v}_i^T \boldsymbol{\phi}_n \}, \qquad i \leq K. \end{aligned} \tag{A.6}$$

If this maximum value is negative, then set $z_{n(K+1)} = 1$ and all other terms to

zeros.

V:

$$\begin{aligned}\log q_V^*(V) =& \langle \log\{\prod_n p(\boldsymbol{z}_n|V,\boldsymbol{\phi}_n)p(V|\beta)\}\rangle + const \\ =& \langle \sum_n \sum_{i=1}^K \{t(i,\boldsymbol{z}_n)(z_{ni}\boldsymbol{v}_i^T\boldsymbol{\phi}_n + \log\sigma(-\boldsymbol{v}_i^T\boldsymbol{\phi}_n))\} + \sum_{i=1}^K \log p(\boldsymbol{v}_i|\beta)\rangle + const.\end{aligned} \tag{A.7}$$

logistic sigmoid function has a lower bound:

$$\sigma(x) \geq \sigma(\varepsilon)\exp\{(x-\varepsilon)/2 - \lambda(\varepsilon)(x^2-\varepsilon^2)\}, \tag{A.8}$$

where

$$\lambda(\varepsilon) = \frac{1}{2\varepsilon}\left[\sigma(\varepsilon) - \frac{1}{2}\right]. \tag{A.9}$$

Substitute $\sigma(-\boldsymbol{v}_i^T\boldsymbol{\phi}_{\boldsymbol{n}})$ with its lower bound in Equation (A.8), we have:

$$\log q_V^*(V) \geq \langle \sum_n \sum_{i=1}^K t(i,\boldsymbol{z}_n)\{(z_{ni}-\frac{1}{2})\boldsymbol{v}_i^T\boldsymbol{\phi}_n - \boldsymbol{v}_i^T\lambda(\varepsilon_{ni})\boldsymbol{\phi}_n\boldsymbol{\phi}_n^T\boldsymbol{v}_i\} - \frac{1}{2}\sum_{i=1}^K \boldsymbol{v}_i^T\beta\boldsymbol{v}_i\rangle + const, \tag{A.10}$$

and then we can write down the lower bound of distribution for each $q_{\boldsymbol{v}_i}(\boldsymbol{v}_i)$ as follows:

$$\begin{aligned}\log q_{\boldsymbol{v}_i}(\boldsymbol{v}_i) \geq& \langle \sum_n t(i,\boldsymbol{z}_n)(z_{ni}-\frac{1}{2})\boldsymbol{v}_i^T\boldsymbol{\phi}_{\boldsymbol{n}} - \\ & \boldsymbol{v}_i^T(\sum_n t(i,\boldsymbol{z}_n)\lambda(\varepsilon_{ni})(\boldsymbol{\phi}_n\boldsymbol{\phi}_n^T))\boldsymbol{v}_i - \boldsymbol{v}_i^T(\frac{\beta}{2})\boldsymbol{v}_i\rangle + const.\end{aligned} \tag{A.11}$$

By completing the square, we can write down the lower bound of the posterior distribution of each $q_{\boldsymbol{v}_i}(\boldsymbol{v}_i)$, which is a Gaussian distribution:

$$q_{\boldsymbol{v}_i}(\boldsymbol{v}_i) \sim \mathcal{N}(\boldsymbol{v}_i|\boldsymbol{m}_{\boldsymbol{v}_i}, \boldsymbol{S}_{\boldsymbol{v}_i}), \tag{A.12}$$

where $\boldsymbol{m}_{\boldsymbol{v}_i}, \boldsymbol{S}_{\boldsymbol{v}_i}$ are given by:

$$\boldsymbol{m}_{\boldsymbol{v}_i} = \boldsymbol{S}_{\boldsymbol{v}_i}(\sum_n t(i,\boldsymbol{z}_n)(z_{ni}-\frac{1}{2})\boldsymbol{\phi}_n), \tag{A.13}$$

$$\boldsymbol{S}_{\boldsymbol{v}_i}^{-1} = \beta I + 2\sum_n t(i,\boldsymbol{z}_n)\lambda(\varepsilon_{ni})(\boldsymbol{\phi}_n\boldsymbol{\phi}_n^T). \tag{A.14}$$

$\boldsymbol{W}$:

$$\begin{aligned}\log q_{\boldsymbol{w}_i}(\boldsymbol{w}_i) &= \langle \log\prod_n\{p(y_n|\boldsymbol{w}_i,z_{ni},\boldsymbol{\phi}_n,\tau_i)\}p(\boldsymbol{w}_i|\alpha)\rangle + const\\ &= \langle\sum_n z_{ni}(\frac{1}{2}\log\tau_i - \frac{1}{2}\log 2\pi - \frac{1}{2}\tau_i(y_n-\boldsymbol{w}_i^T\phi)^2) - \frac{1}{2}\boldsymbol{w}_i^T\alpha_i\boldsymbol{w}_i\rangle + const\\ &= \langle\sum_n z_{ni}\tau_i y_n\boldsymbol{w}_i^T\boldsymbol{\phi}_n - \frac{1}{2}\boldsymbol{w}_i^T(\tau_i\sum_n z_{ni}\boldsymbol{\phi}_n\boldsymbol{\phi}_n^T)\boldsymbol{w}_i\rangle + const.\end{aligned} \tag{A.15}$$

Similarly, $\boldsymbol{w}_i$ follows a normal distribution:

$$q^*_{\boldsymbol{w}_i}(\boldsymbol{w}_i) \sim \mathcal{N}(\boldsymbol{w}_i|\boldsymbol{m}_{\boldsymbol{w}_i},\boldsymbol{S}_{\boldsymbol{w}_i}), \tag{A.16}$$

where $\boldsymbol{m}_{\boldsymbol{w}_i},\boldsymbol{S}_{\boldsymbol{w}_i}$ are given by:

$$\boldsymbol{m}_{\boldsymbol{w}_i} = \boldsymbol{S}_{\boldsymbol{w}_i}\tau_i\sum_n z_{ni}y_n\boldsymbol{\phi}_n, \tag{A.17}$$

$$\boldsymbol{S}_{\boldsymbol{w}_i}^{-1} = \alpha_i I + \tau_i\sum_n z_{ni}\boldsymbol{\phi}_n\boldsymbol{\phi}_n^T. \tag{A.18}$$

$\boldsymbol{\alpha}$:

$$\begin{aligned}\log q^*_{\alpha_i}(\alpha_i) =& \langle\log p(\boldsymbol{w}_i|\alpha_i)p(\alpha_i)\rangle + const\\ =& \langle\log(\sqrt{\frac{\alpha_i}{2\pi}}\exp(-\frac{1}{2}\boldsymbol{w}_i^T\alpha_i\boldsymbol{w}_i)\cdot\frac{1}{\Gamma(\alpha_i)}b^a\alpha^{a-1}\exp(-b\alpha_i))\rangle + const\\ =& \langle\frac{1}{2}\log\alpha_i - \frac{1}{2}\boldsymbol{w}_i^T\alpha_i\boldsymbol{w}_i - \log\Gamma(\alpha_i) + (a-1)\log\alpha_i - b^{\alpha_i}\rangle + const\\ =& -(b+\frac{1}{2}\langle\boldsymbol{w}_i^T\boldsymbol{w}_i\rangle)\alpha_i + (a+\frac{1}{2}-1)\log\alpha_i - \log\Gamma(\alpha_i) + const,\end{aligned} \tag{A.19}$$

where $\langle \boldsymbol{w}_i^T \boldsymbol{w}_i \rangle$ can be further presented by:

$$\begin{aligned} \langle \boldsymbol{w}_i^T \boldsymbol{w}_i \rangle =& tr(\langle \boldsymbol{w}_i^T \boldsymbol{w}_i \rangle) \\ =& \langle tr(\boldsymbol{w}_i^T \boldsymbol{w}_i) \rangle \\ =& tr(\boldsymbol{S}_{\boldsymbol{w}_i} + \boldsymbol{m}_{\boldsymbol{w}_i}^T \boldsymbol{m}_{\boldsymbol{w}_i}). \end{aligned} \tag{A.20}$$

α_i follows a Gamma distribution:

$$\alpha_i \sim Gam(\alpha_i | a + \frac{1}{2}, b + \frac{1}{2} \langle \boldsymbol{w}_i^T \boldsymbol{w}_i \rangle), \tag{A.21}$$

and the expectation of α_i is :

$$\langle \alpha_i \rangle^* = \frac{a + \frac{1}{2}}{b + \frac{1}{2} tr(\boldsymbol{S}_{\boldsymbol{w}_i} + \boldsymbol{m}_{\boldsymbol{w}_i}^T \boldsymbol{m}_{\boldsymbol{w}_i})}. \tag{A.22}$$

$\boldsymbol{\tau}$:

$$\begin{aligned} \log q_{\tau_i}^*(\tau_i) =& \langle \sum_n \log p(y_n | \boldsymbol{\phi}_n, \boldsymbol{w}_i, \tau_i, \boldsymbol{z}_n) p(\tau) \rangle + const \\ =& \langle \sum_n z_{ni} (\frac{1}{2} \log \tau_i - \frac{1}{2} \tau_i (y_n - \boldsymbol{w}_i^T \boldsymbol{\phi}_n)^2) \\ & - \log \Gamma(\tau_i) + a \log b + (a-1) \log \tau_i - b\tau \rangle + const \\ =& \langle (a + \frac{1}{2} \sum_n -1) \log \tau_i - \log \Gamma(\tau_i) - (b + \frac{1}{2} \sum_n z_{ni} (y_n - \boldsymbol{w}_i^T \boldsymbol{\phi}_n)^2 \tau_i) \rangle + const. \end{aligned} \tag{A.23}$$

τ_i follows a Gamma distribution:

$$\tau_i \sim Gam(\tau_i | a + \frac{1}{2} \sum_n z_{ni}, b + \frac{1}{2} \sum_n z_{ni} (y_n - \boldsymbol{w}_i^T \boldsymbol{\phi}_n)^2), \tag{A.24}$$

and the expectation of τ_i is:

$$\langle \tau_i \rangle^* = \frac{a + \frac{1}{2} \sum_n z_{ni}}{b + \frac{1}{2} \sum_n z_{ni} (y_n - \boldsymbol{w}_i^T \boldsymbol{\phi}_n)^2}. \tag{A.25}$$

$\boldsymbol{\beta}$:

$$\begin{aligned}\log q^*_{\beta_i}(\beta_i) =& \langle \log p(\boldsymbol{v}_i|\beta_i)p(\beta_i)\rangle + const \\ =& \langle \log(\sqrt{\frac{\beta_i}{2\pi}}\exp(-\frac{1}{2}\boldsymbol{v}_i^T\beta_i\boldsymbol{v}_i)\cdot\frac{1}{\Gamma(\beta_i)}b^a\beta^{a-1}\exp(-b\beta_i))\rangle + const \\ =& \langle \frac{1}{2}\log\beta_i - \frac{1}{2}\boldsymbol{v}_i^T\beta_i\boldsymbol{v}_i - \log\Gamma(\beta_i) + (a-1)\log\beta_i - b^{\beta_i}\rangle + const \\ =& -(b+\frac{1}{2}\langle\boldsymbol{v}_i^T\boldsymbol{v}_i\rangle)\beta_i + (a+\frac{1}{2}-1)\log\beta_i - \log\Gamma(\beta_i) + const.\end{aligned} \tag{A.26}$$

Distribution and the expectation of β_i can be presented by:

$$\beta_i \sim Gam(\beta_i|a+\frac{1}{2}, b+\frac{1}{2}\langle\boldsymbol{v}_i^T\boldsymbol{v}_i\rangle), \tag{A.27}$$

$$\langle\beta_i\rangle^* = \frac{a+\frac{1}{2}}{b+\frac{1}{2}tr(\boldsymbol{S}_{\boldsymbol{v}_i}+\boldsymbol{m}_{\boldsymbol{v}_i}^T\boldsymbol{m}_{\boldsymbol{v}_i})}. \tag{A.28}$$

$\boldsymbol{\varepsilon}$:

$\boldsymbol{\varepsilon}$ is optimized by M-Step in the variational framework. Refer to conclusion in Equation (5.19), it requires the expectation of posterior distribution $p(\mathcal{X},\mathcal{Y}|\theta)$ with all latent variable fixed. It only depends on the sufficient statistics of $\theta = \{\boldsymbol{\varepsilon}\}$, $\boldsymbol{\varepsilon}$ after the update can be presented by:

$$\begin{aligned}\boldsymbol{\varepsilon} =& \underset{\boldsymbol{\varepsilon}}{argmax}\langle p(\mathcal{X},\mathcal{Y}|\theta)\rangle \\ =& \underset{\boldsymbol{\varepsilon}}{argmax}\langle \log\prod_{i=1}^{K} t(i,z_{ni}) * \sigma(-\boldsymbol{v}_i\boldsymbol{\Phi})\rangle \\ =& \underset{\boldsymbol{\varepsilon}}{argmax}\langle \sum_n t(i,z_{ni})(\log\sigma(\varepsilon_{ni}) - \frac{1}{2}\varepsilon_{ni} - \lambda(\varepsilon_{ni})(\boldsymbol{\phi}_n^T\langle\boldsymbol{v}_i\boldsymbol{v}_i^T\rangle - \varepsilon_{ni}^2))\rangle.\end{aligned} \tag{A.29}$$

set the derivative of above equation w.r.t $\boldsymbol{\varepsilon}$ to 0, we have:

$$\frac{\partial}{\partial\varepsilon_{ni}} = t(i,z_{ni})(\frac{1}{\sigma(\varepsilon_{ni})}\sigma'(\varepsilon_{ni}) - \frac{1}{2} + \sigma(\varepsilon_{ni}) - \frac{1}{2} - \lambda'(\varepsilon_{ni})(\boldsymbol{\phi}_n^T\langle\boldsymbol{v}_i\boldsymbol{v}_i^T\rangle\boldsymbol{\phi}_n - \varepsilon_{ni}^2)) \equiv 0. \tag{A.30}$$

Finally we can get:

$$\begin{aligned}\varepsilon_{ni}^2 &= \boldsymbol{\phi}_n^T \langle \boldsymbol{v}_i \boldsymbol{v}_i^T \rangle \boldsymbol{\phi}_n \\ &= \boldsymbol{\phi}_n^T (\boldsymbol{S}_{\boldsymbol{v}_i} + \boldsymbol{m}_{\boldsymbol{v}_i} \boldsymbol{m}_{\boldsymbol{v}_i}^T) \boldsymbol{\phi}_n.\end{aligned} \tag{A.31}$$

Appendix B

Examples of optimum regularization parameters (λ) in the CNN models for different constituents

Dataset name	Constituent	Concentration range%	Optimum λ
a	Moisture	6 - 16	0.002
b	protein	10 - 60	0.02
c	Ash	0.3 - 1.1	0.0001
d	Water absorption	50 - 70	0.03
e	Protein	7 - 18	0.003
f	Ash	0.3 - 2.0	0.0001

Table B.1: CNN regularization parameters for different datasets. The values of λ were found by grid search.

Bibliography

[1] KH Norris and JR Hart. Principles and methods of measuring moisture in liquids and solids. *Wexler, Reinhold, New York*, pages 19–25, 1965.

[2] GS Birth. A nondestructive technique for detecting internal discolorations in potatoes. *American Potato Journal*, 37(2):53–60, 1960.

[3] JC Cowles. Theory of dual-wavelength spectrophotometry for turbid samples. *Journal of the Optical Society of America*, 55(6):690–692, 1965.

[4] KH Norris and JR Hart. Direct spectrophotometric determination of moisture content of grain and seeds. *Proceedings of the 1963 International Symposium on Humidity and Moisture*, 4:19–25, 1965.

[5] I Bengera and KH Norris. Direct spectrophotometric determination of fat and moisture in meat products. *Journal of Food Science*, 33(1):64–67, 1968.

[6] DR Massie and KH Norris. Spectral reflectance and transmittance properties of grain in the visible and near infrared. *Transactions of the ASAE*, 8(4):598–0600, 1965.

[7] I Landa and KH Norris. Rapid method for the measurement of monochromator parameters in the near infrared. *Applied Spectroscopy*, 33(2):105–107, 1979.

[8] JP Connell and KH Norris. The prediction of the yield of greasy wool by near-infrared reflectance spectroscopy: Part 1: Principles and washed wool. *Textile Research Journal*, 50(6):371–380, 1980.

[9] KH Norris, RF Barnes, JE Moore, and JS Shenk. Predicting forage quality by infrared replectance spectroscopy. *Journal of Animal Science*, 43(4):889–897, 1976.

[10] BG Osborne, S Douglas, and T Fearn. Recent progress in the application of nir to the measurement of quality parameters in flour. In *Progress in Cereal Chemistry and Technology, Proceedings of 7th World Cereal and Bread Congress*, pages 577–581. Elsevier, 1983.

[11] DP Law and R Tkachuk. Near infrared diffuse reflectance spectra of wheat and wheat components. Technical report, 1977.

[12] WF McClure, KH Norris, and WW Weeks. Rapid spectrophotometric analysis of the chemical composition of tobacco: Part 1: Total reducing sugars. *Beiträge zur Tabakforschung/Contributions to Tobacco Research*, 9(1):13–18, 1977.

[13] BG Osborne, T Fearn, AR Miller, and S Douglas. Application of near infrared reflectance spectroscopy to the compositional analysis of biscuits and biscuit doughs. *Journal of the Science of Food and Agriculture*, 35(1):99–105, 1984.

[14] H Martens and T Næs. Multivariate calibration. *Chichester, UK*, 1989.

[15] AE Hoerl and RW Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[16] M Goldstein and AF Smith. Ridge-type estimators for regression analysis. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 284–291, 1974.

[17] T Fearn. A misuse of ridge regression in the calibration of a near infrared reflectance instrument. *Applied Statistics*, pages 73–79, 1983.

[18] AE Hoerl, RW Kennard, and RW Hoerl. Practical use of ridge regression: A challenge met. *Applied Statistics*, pages 114–120, 1985.

[19] P Geladi, D MacDougall, and H Martens. Linearization and scatter-correction for near-infrared reflectance spectra of meat. *Applied Spectroscopy*, 39(3):491–500, 1985.

[20] IA Cowe and JW McNicol. The use of principal components in the analysis of near-infrared spectra. *Applied Spectroscopy*, 39(2):257–266, 1985.

[21] T Næs and H Martens. Principal component regression in nir analysis: viewpoints, background details and selection of components. *Journal of Chemometrics*, 2(2):155–167, 1988.

[22] T Næs, C Irgens, and H Martens. Comparison of linear statistical methods for calibration of NIR instruments. *Applied Statistics*, pages 195–206, 1986.

[23] H Martens, T Karstang, and T Næs. Improved selectivity in spectroscopy by multivariate calibration. *Journal of Chemometrics*, 1(4):201–219, 1987.

[24] H Wold. Path models with latent variables: The NIPALS approach. In *Quantitative Sociology*, pages 307–357. Elsevier, 1975.

[25] S Wold, H Martens, and H Wold. The multivariate calibration problem in chemistry solved by the PLS method. In *Matrix pencils*, pages 286–293. Springer, 1983.

[26] H Martens and SA Jensen. Partial least squares regression: a new two-stage NIR calibration method . *Developments in Food Science*, 1983.

[27] S Wold, A Ruhe, H Wold, and WJ Dunn. The collinearity problem in linear regression. The partial least squares (PLS) approach to generalized inverses . *SIAM Journal on Scientific and Statistical Computing*, 5(3):735–743, 1984.

[28] WF McClure, A Hamid, FG Giesbrecht, and WW Weeks. Fourier analysis enhances NIR diffuse reflectance spectroscopy. *Applied Spectroscopy*, 38(3):322–329, 1984.

[29] AMC Davies and WF McClure. Near infrared analysis of foods. near infrared analysis in the fourier domain with special reference to process control. In *Analytical Proceedings*, volume 22, pages 321–322. Royal Society of Chemistry, 1985.

[30] WF McClure and Anthony MC Davies. Fourier self-deconvolution in the analysis of near infrared spectra of chemically complex samples. *Microchimica Acta*, 94(1-6):93–96, 1988.

[31] PC Williams. Application of near infrared reflectance spectroscopy to analysis of cereal grains and oilseeds. *Cereal Chemistry*, 1975.

[32] RA Stermer, Y Pomeranz, and RJ McGinty. Infrared reflectance spectroscopy for estimation of moisture of whole grain. *Cereal Chemistry*, 54(2):345–351, 1977.

[33] R Tkachuk. Oil and protein analysis of whole rapeseed kernels by near infrared reflectance spectroscopy. *Journal of the American Oil Chemists Society*, 58(8):819–822, 1981.

[34] PC Williams, KH Norris, and DC Sobering. Determination of protein and moisture in wheat and barley by near-infrared transmission. *Journal of Agricultural and Food Chemistry*, 33(2):239–244, 1985.

[35] PC Williams and DC Sobering. Comparison of commercial near infrared transmittance and reflectance instruments for analysis of whole grains and seeds. *Journal of Near Infrared Spectroscopy*, 1(1):25–32, 1993.

[36] BG Osborne, S Douglas, T Fearn, and KH Willis. The development of universal calibrations for measurement of protein and moisture in uk homegrown wheat by near-infrared reflectance analysis. *Journal of the Science of Food and Agriculture*, 33(8):736–740, 1982.

[37] M Manley, L Van Zyl, and BG Osborne. Using fourier transform near infrared spectroscopy in determining kernel hardness, protein and mois-

ture content of whole wheat flour. *Journal of Near Infrared Spectroscopy*, 10(1):71–76, 2002.

[38] H Pettersson and L Åberg. Near infrared spectroscopy for determination of mycotoxins in cereals. *Food Control*, 14(4):229–232, 2003.

[39] CA Watson, D Carville, E Dikeman, G Daigger, and GD Booth. Evaluation of two infrared instruments for determining protein content of hard red winter wheat. *Cereal Chemistry (USA)*, 1976.

[40] BG Osborne, S Douglas, and T Fearn. The application of near infrared reflectance analysis to rapid flour testing. *International Journal of Food Science & Technology*, 17(3):355–363, 1982.

[41] GA Hareland. Evaluation of flour particle size distribution by laser diffraction, sieve analysis and near-infrared reflectance spectroscopy. *Journal of Cereal Science*, 20(2):183–190, 1994.

[42] BG Osborne and T Fearn. Collaborative evaluation of universal calibrations for the measurement of protein and moisture in flour by near infrared reflectance. *International Journal of Food Science & Technology*, 18(4):453–460, 1983.

[43] VR Diachuk, E Hamilton, N Savchuk, and SS Jackel. Bakery flour control using near-infrared reflectance analysis. *Bakers Digest*, 55(5):72, 1981.

[44] M Iwamoto, C Kwang, T Suzuki, and J Uozumi. Near infrared reflectance analysis for determing moisture, protein and ash contents in home-grown wheat flours. *Nippon Shokuhin Kogyo Gakkaishi*, 31(1):50–53, 1984.

[45] C Miralbes. Quality control in the milling industry using near infrared transmittance spectroscopy. *Food Chemistry*, 88(4):621–628, 2004.

[46] Floyd E Dowell, James E Throne, and James E Baker. Automated nondestructive detection of internal insect infestation of wheat kernels by using

near-infrared reflectance spectroscopy. *Journal of Economic Entomology*, 91(4):899–904, 1998.

[47] J Perez-Mendoza, JE Throne, FE Dowell, and JE Baker. Detection of insect fragments in wheat flour by near-infrared spectroscopy. *Journal of Stored Products Research*, 39(3):305–312, 2003.

[48] CB Singh, DS Jayas, J Paliwal, and NDG White. Detection of insect-damaged wheat kernels using near-infrared hyperspectral imaging. *Journal of Stored Products Research*, 45(3):151–158, 2009.

[49] CB Singh, DS Jayas, J Paliwal, and NDG White. Identification of insect-damaged wheat kernels using short-wave near-infrared hyperspectral and digital colour imaging. *Computers and Electronics in Agriculture*, 73(2):118–125, 2010.

[50] FE Barton and D Burdick. Prediction of forage quality with nir reflectance spectroscopy. In *Proceedings of the XVI International Grassland Congress: Held at Lexington, Kentucky, USA June 15-24, 1981/edited by J. Allan Smith and Virgil W. Hays*. Boulder, Colo.: Westview Press, 1983.

[51] JS Shenk and RF Barnes. Current status of infrared reflectance "rapid evaluation of forage quality, sheep". In *Proceedings Southern Pasture and Forage Crop Improvement Conference*, 1977.

[52] GE Counts and HD Radloff. The potential of infrared analysis in forage evaluation. In *Proceedings of the Annual Meeting. American Society for Animal Science Western Section*, 1979.

[53] JE Winch and H Major. Predicting nitrogen and digestibility of forages using near infrared reflectance photometry. *Canadian Journal of Plant Science*, 61(1):45–51, 1981.

[54] FE Barton and D Burdick. Preliminary study on the analysis of forages with a filter-type near-infrared reflectance spectrometer. *Journal of Agricultural and Food Chemistry*, 27(6):1248–1252, 1979.

[55] SL Fales and DG Cummins. Reducing moisture-induced error associated with measuring forage quality using near infrared reflectance 1. *Agronomy Journal*, 74(3):585–588, 1982.

[56] DJ Minson, KL Butler, N Grummitt, and DP Law. Bias when predicting crude protein, dry matter digestibility and voluntary intake of tropical grasses by near-infrared reflectance. *Animal Feed Science and Technology*, 9(3):221–237, 1983.

[57] I Bengera and KH Norris. Influence of fat concentration on absorption spectrum of milk in near-infrared region. *Israel Journal of Agricultural Research*, 18(3):117, 1968.

[58] P Casado, C Blanco, and A Pozas. Analysis of dried milk by diffuse reflectance spectroscopy. *Rev. Esp. Lecheria*, 108:97, 1978.

[59] J Vilder and R Bossuyt. Practical experiences with an infraalyzer 400 in determining the water, protein and fat content of milk powder. *Milchwissenschaft*, 1983.

[60] RJ Baer, JF Frank, and M Loewenstein. Compositional analysis of nonfat dry milk by using near infrared diffuse reflectance spectroscopy. *Journal Association of Official Analytical Chemists*, 66(4):858–863, 1983.

[61] R Frankhuizen and NG VanderVeen. Determination of major and minor constituents in milk powders and cheese by near infra-red reflectance spectroscopy. *Netherlands Milk and Dairy Journal*, 1985.

[62] G Downey, P Robert, D Bertrand, and PM Kelly. Classification of commercial skim milk powders according to heat treatment using factorial discrim-

inant analysis of near-infrared reflectance spectra. *Applied Spectroscopy*, 44(1):150–155, 1990.

[63] P Robert, D Bertrand, MF Devaux, and R Grappin. Multivariate analysis applied to near-infrared spectra of milk. *Analytical Chemistry*, 59(17):2187–2191, 1987.

[64] MF Laporte and P Paquin. Near-infrared analysis of fat, protein, and casein in cow's milk. *Journal of Agricultural and Food Chemistry*, 47(7):2600–2605, 1999.

[65] R Giangiacomo, D Torreggiani, JF Frank, M Loewenstein, and GS Birth. Analysis of blue cheese using infrared reflectance spectroscopy.[paper presented at a conference]. *Attidell'Istituto Sperimentale per la Valorizzazione Tecnologica dei Prodotti Agricoli, Milano*, 1979.

[66] JF Frank and GS Birth. Application of near infrared reflectance spectroscopy to cheese analysis. *Journal of Dairy Science*, 65(7):1110–1116, 1982.

[67] L Pillonel, W Luginbühl, D Picque, E Schaller, R Tabacchi, and J Bosset. Analytical methods for the determination of the geographic origin of emmental cheese: mid-and near-infrared spectroscopy. *European Food Research and Technology*, 216(2):174–178, 2003.

[68] LK Sørensen and R Jepsen. Assessment of sensory properties of cheese by near-infrared spectroscopy. *International Dairy Journal*, 8(10-11):863–871, 1998.

[69] I González-Martín, JM Hernández-Hierro, R Morón-Sancho, J Salvador-Esteban, A Vivar-Quintana, and I Revilla. Determination of the percentage of milk (cow's, ewe's and goat's) in cheeses with different ripening times using near infrared spectroscopy technology and a remote reflectance fibre-optic probe. *Analytica Chimica Acta*, 604(2):191–196, 2007.

[70] M Pouliot, P Paquin, R Martel, SF Gauthier, and Y Pouliot. Whey changes during processing determined by near infrared spectroscopy. *Journal of Food Science*, 62(3):475–479, 1997.

[71] L Nørgaard, MT Hahn, LB Knudsen, IA Farhat, and SB Engelsen. Multivariate near-infrared and raman spectroscopic quantifications of the crystallinity of lactose in whey permeate powder. *International Dairy Journal*, 15(12):1261–1270, 2005.

[72] P Berzaghi, A Dalle Zotte, LM Jansson, and I Andrighetto. Near-infrared reflectance spectroscopy as a method to predict chemical composition of breast meat and discriminate between different n-3 feeding sources. *Poultry Science*, 84(1):128–136, 2005.

[73] S Anderson. Determination of fat, moisture, and protein in meat and meat products by using the foss foodscan near-infrared spectrophotometer with foss artificial neural network calibration model and associated database: collaborative study. *Journal of AOAC International*, 90(4):1073–1083, 2007.

[74] R Rødbotten, BN Nilsen, and KI Hildrum. Prediction of beef quality attributes from early post mortem near infrared reflectance spectra. *Food Chemistry*, 69(4):427–436, 2000.

[75] DE Chan, PN Walker, and EW Mills. Prediction of pork quality characteristics using visible and near–infrared spectroscopy. *Transactions of the ASAE*, 45(5):1519, 2002.

[76] S Andrés, I Murray, EA Navajas, AV Fisher, NR Lambe, and L Bünger. Prediction of sensory characteristics of lamb meat samples by near infrared reflectance spectroscopy. *Meat science*, 76(3):509–516, 2007.

[77] CI Gerhausser and K Kovar. Strategies for constructing near-infrared spectral libraries for the identification of drug substances. *Applied Spectroscopy*, 51(10):1504–1510, 1997.

[78] NK Ebube, SS Thosar, RA Roberts, MS Kemper, R Rubinovitz, DL Martin, GE Reier, TA Wheatley, and AJ Shukla. Application of near-infrared spectroscopy for nondestructive analysis of avicel® powders and tablets. *Pharmaceutical Development and Technology*, 4(1):19–26, 1999.

[79] K Krämer and S Ebel. Application of NIR reflectance spectroscopy for the identification of pharmaceutical excipients. *Analytica Chimica Acta*, 420(2):155–161, 2000.

[80] M Andre. Multivariate analysis and classification of the chemical quality of 7-aminocephalosporanic acid using near-infrared reflectance spectroscopy. *Analytical Chemistry*, 75(14):3460–3467, 2003.

[81] R De Maesschalck and T Van den Kerkhof. Implementation of a simple semi-quantitative near-infrared method for the classification of clinical trial tablets. *Journal of Pharmaceutical and Biomedical Analysis*, 37(1):109–114, 2005.

[82] N Sondermann and K Kovar. Screening experiments of ecstasy street samples using near infrared spectroscopy. *Forensic Science International*, 106(3):147–156, 1999.

[83] H Hausdorff. Analysis of polymers by infrared spectroscopy. In *Analytical Chemistry*, volume 23, pages 683–683, 1951.

[84] HE Howell and JR Davis. Qualitative identification of polymeric materials using near-infrared spectroscopy. ACS Publications, 1993.

[85] W Camacho and S Karlsson. Nir, dsc, and ftir as quantitative methods for compositional analysis of blends of polymers obtained from recycled mixed plastic waste. *Polymer Engineering & Science*, 41(9):1626–1635, 2001.

[86] A Cherfi and G Févotte. On-line conversion monitoring of the solution polymerization of methyl methacrylate using near-infrared spectroscopy. *Macromolecular Chemistry and Physics*, 203(9):1188–1193, 2002.

[87] E Minopoulou, E Dessipri, Georgios D Chryssikos, V Gionis, A Paipetis, and C Panayiotou. Use of NIR for structural characterization of urea–formaldehyde resins. *International Journal of Adhesion and Adhesives*, 23(6):473–484, 2003.

[88] LA Marquardt, MA Arnold, and GW Small. Near-infrared spectroscopic measurement of glucose in a protein matrix. *Analytical Chemistry*, 65(22):3271–3278, 1993.

[89] CH Fischbacher, KU Jagemann, K Danzer, UA Müller, L Papenkordt, and J Schüler. Enhancing calibration models for non-invasive near-infrared spectroscopical blood glucose determination. *Fresenius' Journal of Analytical Chemistry*, 359(1):78–82, 1997.

[90] K Maruo, M Tsurugi, J Chin, T Ota, H Arimoto, Y Yamada, M Tamura, M Ishii, and Y Ozaki. Noninvasive blood glucose assay using a newly developed near-infrared system. *IEEE Journal of Selected Topics in Quantum Electronics*, 9(2):322–330, 2003.

[91] JJ Baraga, MS Feld, and RP Rava. Rapid near-infrared raman spectroscopy of human tissue with a spectrograph and ccd detector. *Applied Spectroscopy*, 46(2):187–190, 1992.

[92] DT Delpy and M Cope. Quantification in tissue near–infrared spectroscopy. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 352(1354):649–659, 1997.

[93] SJ Gaffey. Spectral reflectance of carbonate minerals in the visible and near infrared (0.35–2.55 um): Anhydrous carbonate minerals. *Journal of Geophysical Research: Solid Earth*, 92(B2):1429–1440, 1987.

[94] JM Andres and MT Bona. Analysis of coal by diffuse reflectance near-infrared spectroscopy. *Analytica Chimica Acta*, 535(1-2):123–132, 2005.

[95] D Kim, J Lee, and J Kim. Application of near infrared diffuse reflectance spectroscopy for on-line measurement of coal properties. *Korean Journal of Chemical Engineering*, 26(2):489–495, 2009.

[96] MJ Morra, MH Hall, and LL Freeborn. Carbon and nitrogen analysis of soil fractions using near-infrared reflectance spectroscopy. *Soil Science Society of America Journal*, 55(1):288–291, 1991.

[97] E Ben-Dor and A Banin. Near-infrared analysis as a rapid method to simultaneously evaluate several soil properties. *Soil Science Society of America Journal*, 59(2):364–372, 1995.

[98] C Chang, David A Laird, MJ Mausbach, and CR Hurburgh. Near-infrared reflectance spectroscopy–principal components regression analyses of soil properties. *Soil Science Society of America Journal*, 65(2):480–490, 2001.

[99] RAV Rossel, DJJ Walvoort, AB McBratney, LJ Janik, and JO Skjemstad. Visible, near infrared, mid infrared or combined diffuse reflectance spectroscopy for simultaneous assessment of various soil properties. *Geoderma*, 131(1-2):59–75, 2006.

[100] B Stenberg, RAV Rossel, AM Mouazen, and J Wetterlind. Visible and near infrared spectroscopy in soil science. In *Advances in agronomy*, volume 107, pages 163–215. Elsevier, 2010.

[101] S Tsuchikawa. A review of recent near infrared research for wood and paper. *Applied Spectroscopy Reviews*, 42(1):43–71, 2007.

[102] Yves Roggo, Pascal Chalus, Lene Maurer, Carmen Lema-Martinez, Aurélie Edmond, and Nadine Jent. A review of near infrared spectroscopy and chemometrics in pharmaceutical technologies. *Journal of Pharmaceutical and Biomedical Analysis*, 44(3):683–700, 2007.

[103] HB Ding and RJ Xu. Differentiation of beef and kangaroo meat by visible/near-infrared reflectance spectroscopy. *Journal of Food Science*, 64(5):814–817, 1999.

[104] RM Balabin and RZ Safieva. Gasoline classification by source and type based on near infrared (nir) spectroscopy data. *Fuel*, 87(7):1096–1101, 2008.

[105] T Næs, T Isaksson, T Fearn, and T Davies. *A User Friendly Guide to Multivariate Calibration and Classification*. 2002.

[106] DK Pedersen, H Martens, JP Nielsen, and SB Engelsen. Near-infrared absorption and scattering separated by extended inverted signal correction (eisc): analysis of near-infrared transmittance spectra of single wheat seeds. *Applied Spectroscopy*, 56(9):1206–1214, 2002.

[107] V Vapnik. *The Nature of Statistical Learning Theory*. Springer science & business media, 2013.

[108] Y Xu, S Zomer, and RG Brereton. Support vector machines: a recent method for classification in chemometrics. *Critical Reviews in Analytical Chemistry*, 36(3-4):177–188, 2006.

[109] JAK Suykens, T Van Gestel, J De Brabanter, B De Moor, J Vandewalle, JAK Suykens, and T Van Gestel. *Least Squares Support Vector Machines*, volume 4. World Scientific, 2002.

[110] RP Cogdill and P Dardenne. Least-squares support vector machines for chemometrics: an introduction and evaluation. *Journal of Near Infrared Spectroscopy*, 12(2):93–100, 2004.

[111] CE Rasmussen. Gaussian processes in machine learning. In *Advanced Lectures on Machine Learning*, pages 63–71. Springer, 2004.

[112] CE Rasmussen and CKI Williams. *Gaussian Processes For Machine Learning*, volume 1. MIT press Cambridge, 2006.

[113] T Chen, J Morris, and E Martin. Gaussian process regression for multivariate spectroscopic calibration. *Chemometrics and Intelligent Laboratory Systems*, 87(1):59–71, 2007.

[114] T Chen and B Wang. Bayesian variable selection for gaussian process regression: Application to chemometric calibration of spectrometers. *Neurocomputing*, 73(13):2718–2726, 2010.

[115] C Cui and T Fearn. Comparison of partial least squares regression, least squares support vector machines, and gaussian process regression for a near infrared calibration. *Journal of Near Infrared Spectroscopy*, 25(1):5–14, 2017.

[116] CE Rasmussen. *Gaussian Process For Machine Learning*. The MIT Press, 2006.

[117] K Pelckmans, JAK Suykens, T Van Gestel, J De Brabanter, L Lukas, B Hamers, B De Moor, and J Vandewalle. Ls-svmlab: a matlab/c toolbox for least squares support vector machines. *Tutorial. KULeuven-ESAT. Leuven, Belgium*, 2002.

[118] CE Rasmussen and H Nickisch. Gaussian processes for machine learning (gpml) toolbox. *The Journal of Machine Learning Research*, 11:3011–3015, 2010.

[119] T Fearn, D Pérez-Marín, A Garrido-Varo, and JE Guerrero-Ginel. Inverse, classical, empirical and non-parametric calibrations in a bayesian framework. *Journal of Near Infrared Spectroscopy*, 18(1):27, 2010.

[120] G Wahba. *Spline Models For Observational Data*, volume 59. Siam, 1990.

[121] MI Jordan and RA Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6(2):181–214, 1994.

[122] L Breiman, J Friedman, CJ Stone, and RA Olshen. *Classification and Regression Trees*. CRC press, 1984.

[123] JH Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, pages 1–67, 1991.

[124] L Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[125] SR Waterhouse, D MacKay, and AJ Robinson. Bayesian methods for mixtures of experts. In *Advances in Neural Information Processing Systems*, pages 351–357, 1996.

[126] N Ueda and Z Ghahramani. Bayesian model search for mixture models based on optimizing variational bounds. *Neural Networks*, 15(10):1223–1241, 2002.

[127] CM Bishop and M Svenskn. Bayesian hierarchical mixtures of experts. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, pages 57–64. Morgan Kaufmann Publishers Inc., 2002.

[128] WS Cleveland and SJ Devlin. Locally weighted regression: an approach to regression analysis by local fitting. *Journal of the American Statistical Association*, 83(403):596–610, 1988.

[129] T Næs, T Isaksson, and B Kowalski. Locally weighted regression and scatter correction for near-infrared reflectance data. *Analytical Chemistry*, 62(7):664–673, 1990.

[130] C Cui and T Fearn. Hierarchical mixture of linear regressions for multivariate spectroscopic calibration: An application for NIR calibration. *Chemometrics and Intelligent Laboratory Systems*, 2018.

[131] FV Jensen. *An Introduction to Bayesian Networks*, volume 210. UCL press London, 1996.

[132] CM Bishop. *Pattern Recognition and Machine Learning*. springer, 2006.

[133] M Sahani. Lecture notes distributed in probabilistic and unsupervised learning. available from http://www.gatsby.ucl.ac.uk/teaching/courses. 2016.

[134] TS Jaakkola and MI Jordan. Bayesian parameter estimation via variational methods. *Statistics and Computing*, 10(1):25–37, 2000.

[135] M Everingham, L Van Gool, CKI Williams, J Winn, and A Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[136] O Russakovsky, J Deng, H Su, J Krause, S Satheesh, S Ma, Z Huang, A Karpathy, A Khosla, M Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[137] R Collobert and J Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 160–167. ACM, 2008.

[138] Y Zheng, Q Liu, E Chen, Y Ge, and JL Zhao. Time series classification using multi-channels deep convolutional neural networks. In *International Conference on Web-Age Information Management*, pages 298–310. Springer, 2014.

[139] JR Long, VG Gregoriou, and PJ Gemperline. Spectroscopic calibration and quantitation using artificial neural networks. *Analytical Chemistry*, 62(17):1791–1797, 1990.

[140] J Zupan and J Gasteiger. Neural networks: A new method for solving chemical problems or just a passing phase? *Analytica Chimica Acta*, 248(1):1–30, 1991.

[141] W Wu, B Walczak, DL Massart, S Heuerding, F Erni, IR Last, and KA Prebble. Artificial neural networks in classification of nir spectral data: design of the training set. *Chemometrics and Intelligent Laboratory Systems*, 33(1):35–46, 1996.

[142] Z Boger. Selection of quasi-optimal inputs in chemometrics modeling by artificial neural network analysis. *Analytica Chimica Acta*, 490(1):31–40, 2003.

[143] PA Jansson. Neural networks: An overview. *Analytical chemistry*, 63(6):357A–362A, 1991.

[144] JA Burns and GM Whitesides. Feed-forward neural networks in chemistry: mathematical systems for classification and pattern recognition. *Chemical Reviews*, 93(8):2583–2601, 1993.

[145] DA Cirovic. Feed-forward artificial neural networks: applications to spectroscopy. *TrAC Trends in Analytical Chemistry*, 16(3):148–155, 1997.

[146] F Marini, R Bucci, AL Magrì, and AD Magrì. Artificial neural networks in chemometrics: History, examples and perspectives. *Microchemical Journal*, 88(2):178–185, 2008.

[147] S Malek, F Melgani, and Y Bazi. One-dimensional convolutional neural networks for spectroscopic signal regression. *Journal of Chemometrics*, 2017.

[148] EJ Bjerrum, M Glahder, and T Skov. Data augmentation of spectral data for convolutional neural network (cnn) based deep chemometrics. *ArXiv Preprint ArXiv:1710.01927*, 2017.

[149] C Cui and T Fearn. Modern practical convolutional neural networks for multivariate regression: Applications to nir calibration. *Chemometrics and Intelligent Laboratory Systems*, 2018.

[150] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

[151] Y LeCun, L Bottou, GB Orr, and KR Müller. Efficient backprop. pages 9–48. Springer, 2012.

[152] A Krizhevsky, I Sutskever, and GE Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[153] D Clevert, T Unterthiner, and S Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *ArXiv Preprint ArXiv:1511.07289*, 2015.

[154] N Srivastava, GE Hinton, A Krizhevsky, I Sutskever, and R Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[155] S Wager, S Wang, and PS Liang. Dropout training as adaptive regularization. In *Advances in Neural Information Processing Systems*, pages 351–359, 2013.

[156] Y LeCun, Y Bengio, and G Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[157] D Kingma and J Ba. Adam: A method for stochastic optimization. *ArXiv Preprint ArXiv:1412.6980*, 2014.

[158] T Schaul, I Antonoglou, and D Silver. Unit tests for stochastic optimization. *ArXiv Preprint ArXiv:1312.6055*, 2013.

[159] A Krizhevsky. One weird trick for parallelizing convolutional neural networks. *ArXiv Preprint ArXiv:1404.5997*, 2014.

[160] Eric P Xing, Michael I Jordan, Stuart J Russell, and Andrew Y Ng. Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, pages 521–528, 2003.

[161] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[162] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[163] Zhen Zuo, Bing Shuai, Gang Wang, Xiao Liu, Xingxing Wang, Bing Wang, and Yushi Chen. Convolutional recurrent neural networks: Learning spatial dependencies for image representation. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 18–26, 2015.

[164] Pedro HO Pinheiro and Ronan Collobert. Recurrent convolutional neural networks for scene labeling. In *31st International Conference on Machine Learning (ICML)*, number EPFL-CONF-199822, 2014.

[165] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. Convolutional recurrent neural networks for music classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2392–2396. IEEE, 2017.