Fast method for GA-PLS with simultaneous feature selection and identification of optimal preprocessing technique for datasets with many observations

Petter Stefansson | Kristian H. Liland | Thomas Thiis | Ingunn Burud

Abstract
A fast and memory-efficient new method for performing genetic algorithm partial least squares (GA-PLS) on spectroscopic data preprocessed in multiple different ways is presented. The method, which is primarily intended for datasets containing many observations, involves preprocessing a spectral dataset with several different techniques and concatenating the different versions of the data horizontally into a design matrix \( X \) which is both tall and wide. The large matrix is then condensed into a substantially smaller covariance matrix \( X'X \) whose resulting size is unrelated to the number of observations in the dataset, i.e. the height of \( X \). It is demonstrated that the smaller covariance matrix can be used to efficiently calibrate partial least squares (PLS) models containing feature selections from any of the involved preprocessing techniques. The method is incorporated into GA-PLS and used to evolve variable selections for a set of different preprocessing techniques concurrently within a single algorithm. This allows a single instance of GA-PLS to determine which preprocessing technique, within the set of considered methods, is best suited for the spectroscopic dataset. Additionally, the method allows feature selections to be evolved containing variables from a mixture of different preprocessing techniques. The benefits of the introduced GA-PLS technique can be summarized as threefold: (1) for datasets with many observations, the proposed method is substantially faster compared to conventional GA-PLS implementations based on NIPALS, SIMPLS, etc. (2) using a single GA-PLS automatically reveals which of the considered preprocessing techniques results in the lowest model error. (3) it allows the exploration of highly complex solutions composed of features preprocessed using various techniques.

KEYWORDS
GA-PLS, Genetic algorithms, Partial least squares, Feature selection, Preprocessing

1 | INTRODUCTION
Multivariate calibration on spectroscopic near-infrared (NIR) data typically involves overcoming two main challenges: choice of spectral preprocessing method and selection of relevant wavelengths to include in the model. The goal of...
spectral preprocessing is to remove unwanted noise from the measured spectral signal (e.g. light scattering due to physical phenomena), to emphasize features related to the specific chemometric problem at hand and to improve the comparability between samples.\(^1\) The purpose of wavelength selection is to limit the spectral region used by the model to only areas relevant for modeling the analyte of interest; thus enabling faster and more accurate predictions whilst in addition providing a better understanding of the underlying process that generated the data.\(^2\) In many real-world circumstances, a priori information regarding what constitutes a suitable wavelength selection and/or preprocessing technique is not available. To address the challenge of wavelength selection, a myriad of feature selection algorithms can be found suggested in chemometric literature such as simulated annealing, step-wise methods, interval PLS (iPLS) and genetic algorithms (GA).\(^3\)-\(^5\) In the realm of NIR spectroscopy, genetic algorithms have been extensively used for several decades and proven to be a well suited strategy for the selection of relevant wavelengths when used together with partial least squares (PLS) regression\(^6\)-\(^9\) to form a particular niche of GA commonly abbreviated as GA-PLS. Genetic algorithms operate on a population of potential solutions to an optimization task; a population which is improved upon over consecutive generations in an attempt to mimic natural evolution. Using a population of candidate solutions to an optimization task has several advantages. For instance, it reduces the risk of the search getting stuck in a local optimum and allows for a better exploration of the solution space.\(^10\) One of the main drawbacks of GA-PLS is that the computational cost of evaluating the performance of each candidate solution in the population is often high—making the algorithm highly time consuming. In the interest of saving time it is therefore tempting to use a small population size when running a GA, it has however been demonstrated\(^11\) that using a small population size can greatly lower the success rate of genetic algorithms. If one wishes to explore how well various different preprocessing methods perform together with GA-PLS, the computational cost of doing so further increases linearly with every preprocessing technique added to the search.

Recently, a new method for calibrating multiple PLS models using different variable subsets of the same dataset was discovered\(^12\) which greatly reduces the computational cost of performing PLS feature selection on strongly overdetermined, i.e. ‘long and thin’, regression problems. The computational speedup of the method introduced by Stefansson et al.\(^12\) is accomplished by deriving the covariance matrix of any variable subset directly from a covariance matrix calculated using the full set of variables in the dataset; thereby bypassing the need to calculate the covariance for each individual subset. The derived covariance matrices for the various feature subsets are then used to estimate PLS regression coefficients using kernel PLS at a low computational cost. In this study, the method introduced in\(^12\) will be used in the context of GA-PLS and expanded upon to demonstrate that the method not only is capable of reducing the run-time of a conventional GA-PLS, but can also be used to efficiently incorporate multiple different preprocessing techniques within a single population of a GA-PLS. The candidate solutions of the population can then either be restricted to evolve solely within their own preprocessing-specific region of the solution space, which produces the equivalent effect of running multiple genetic algorithms in parallel. Or, the search space region in which candidate solutions are permitted to operate in can be unrestricted, thereby enabling models to evolve containing features from a mixture of different preprocessing techniques.

To experimentally demonstrate that our suggested methodology works in practice, the proposed concept will be applied to two hyperspectral datasets containing NIR and vis-NIR spectra and used to perform simultaneous feature selection and identification of optimal preprocessing method. To quantify the computational efficiency gains offered by the method, the introduced technique will be benchmarked against other GA-PLS implementations using well-established PLS algorithms (SIMPLS, NIPALS, Bidia2 and modified kernel algorithm #2).

The remaining part of this paper is organized as follows: in section 2.1 a brief overview of how conventional GA-PLS algorithms work is given, which will serve as a primer leading up to the new modified/extended GA-PLS concept outlined in section 2.2. In this section it is described how a GA-PLS can be implemented in order to efficiently operate on multiple preprocessing methods simultaneously using the technique from.\(^12\) In section 3 a summary of the preprocessing methods and datasets used to evaluate the proposed algorithm is given as well as a description of the computational efficiency benchmarks performed. The results of which are presented in section 4.

2 | METHOD

2.1 | Overview of GA-PLS

The conventional way of representing a candidate solution to a feature selection problem using GA-PLS is by a Boolean vector of the same length as the total number of variables in the dataset. Such vectors are generally referred to as
chromosomes. Each Boolean value of the vector corresponds to a variable being either excluded from the feature selection, 0, or included, 1. Many different chromosomes placed together in a Boolean matrix constitutes a population. The Boolean values of each solution are typically initialized randomly at the start of the algorithm. The feature subset specified by each chromosome of the population is then used to calibrate one or more PLS regression models on the dataset. The resulting regression coefficients are used to model the target response of the dataset, and the error associated with the model's predictions are reduced into a statistical measure of the feature subset's performance, such as the cross-validated root-mean-squared error, RMSEcv. The cross-validated performance measure of each chromosome is then converted into a fitness value, which is a scalar value that communicates the quality of an encoded solution to the GA and defines what improvement means. How well a chromosome translates into a fitness value, i.e. how well it solves the optimization task compared to all the other candidate solutions of the population, determines how it is treated by the GA-PLS in the steps to come, where fitter individuals are favored compared to individuals with a lower fitness. The bias within the generational cycle of a genetic algorithm towards favoring solutions with high fitness is typically realized using two stages of selection: parent selection and survivor selection. In the parent selection stage it is decided which individuals are used to generate new offspring solutions which will drive evolution forward. In the survivor selection stage, all individuals of the population compete for survival into the next generation. If the fitness value of a chromosome is low compared to the rest of the population, it is more likely to be discarded from the population. In the interest of conciseness the mechanics of the different selection procedures will not be explained here, for an excellent overview of some commonly used selection operators the reader is referred to. In addition to favoring individuals already known to have a high fitness through the selection operators, genetic algorithms have the ability to generate new–potentially improved–solutions to an optimization task by combining and/or refining already successful chromosomes to form new ones. This is achieved using a set of stochastic variation operators: crossover and mutation. The crossover operator takes a pair of parent chromosomes as input and produces one or several offspring chromosomes as output, analogously to how genes from two parents are combined to generate children in nature. An example of a frequently used crossover operator in GA-PLS is 2-point crossover, which is performed by randomly selecting two points along two parent chromosomes and letting the resulting offspring chromosomes alternately inherit sections from their parents between the crossover points. The second variation operator, the mutation, is a unary process that takes a single chromosome as input and outputs a slightly altered version of the same chromosome which replaces the original in the population. In GA-PLS, bitwise mutation is commonly used. In bitwise mutation the Boolean value of each element within a chromosome is flipped—thereby turning a value of 0 into 1 and vice versa—according to a mutation probability \( p_m \). Typically \( p_m \) is conservatively set to a small number since a too high value could have a destructive impact on the evolutionary progress. Placing all the mentioned components together in the order illustrated by the flowchart in Figure 1 forms a conventional PLS-GA.

2.2 Extending/modifying GA-PLS to efficiently operate on multiple versions of the same data with kernel PLS

Certain kernel PLS algorithms, such as the modified kernel algorithm \#2, base their regression coefficient calculations on the covariance matrices of the dataset, \( \mathbf{X}^T \mathbf{X} \) and \( \mathbf{X}^T \mathbf{y} \), rather than the \( \mathbf{X} \) and \( \mathbf{y} \) data itself. For strongly overdetermined systems, kernel algorithms can vastly reduce the computational time required to do PLS compared to traditional well-established algorithms such as NIPALS or SIMPLS. Calculating the covariance is however still a costly operation for large \( \mathbf{X} \) matrices. In a previous paper, Stefansson et al. showed that when performing subset selection, the computationally costly \( \mathbf{X}^T \mathbf{X} \) and \( \mathbf{X}^T \mathbf{y} \), rather than the \( \mathbf{X} \) covariance calculations required for kernel PLS only needs to be established once using the full set of variables within \( \mathbf{X} \). The covariance matrix that any combinatorial column subset of \( \mathbf{X} \) would have resulted in, had it been separately calculated, can then be retrieved from the full covariance matrix without further computations simply by indexing into it. The greater the number of evaluated feature subsets become, the larger the efficiency gains of this technique becomes. When evaluating the performance of an entire population in GA-PLS, this technique enables all the individuals of a population to share the same \( \mathbf{X}^T \mathbf{X} \) and \( \mathbf{X}^T \mathbf{y} \), rather than the \( \mathbf{X} \) covariance matrices as a basis for their regression coefficient estimations. Sharing covariance matrices between multiple feature subsets has shown to greatly decrease the computational time required to estimate PLS regression coefficients under many circumstances. Additionally, the full covariance matrix \( \mathbf{X}^T \mathbf{X} \) will be of dimension \( n \times n \), where \( n \) denotes the number of columns in \( \mathbf{X} \). For strongly overdetermined systems where the number of rows \( m \) in \( \mathbf{X} \) is substantially greater than the number of columns \( m \gg n \), this means that the memory required to store \( \mathbf{X}^T \mathbf{X} \) is greatly reduced compared to storing \( \mathbf{X} \) itself. This in
turn makes the kernel PLS-based indexing technique highly amendable to parallel GPU implementations, where the available memory is often relatively low. In order to derive the covariance matrix of a feature subset from a full $X^T X$ matrix, a Boolean chromosome of length $n$ can simply be aligned along both the rows and columns of the full $X^T X$ matrix, whereupon only elements intersecting with 1 in both dimensions are extracted. The submatrix one receives from this extraction will then be identical to the covariance matrix that would be obtained had the covariance matrix been calculated using only the column subset in X specified by the same chromosome. The same concept also holds true for $X^T y$, except only one dimension is involved when extracting elements intersecting with 1. This indexing concept is visually exemplified using two small chromosomes and a dummy $X^T X$ matrix in Figure 2. For a more thorough description of the indexing technique along with a MATLAB implementation of it, the reader is referred to.\textsuperscript{12}

Considering that evaluating the fitness of a GA-PLS population tends to be the overwhelmingly most time-consuming step within the evolutionary cycle of the algorithm, and that forming the covariance matrix $X^T X$ in turn tends to be the most computationally costly part of a kernel PLS algorithm, the indexing technique illustrated in Figure 2 can alone substantially increase the efficiency of a GA-PLS.\textsuperscript{12} However, since it has been shown that an $X^T X$ matrix can be indexed into using the simple technique illustrated in Figure 2 in order to obtain the covariance matrix related to any subset of columns in X, it naturally follows that if X is widened by adding additional columns to it, any covariance matrix containing these added features can also be retroactively retrieved from the resulting $X^T X$ matrix. By treating a spectroscopic data matrix X with multiple different preprocessing techniques and horizontally concatenating these matrices into one wide matrix:

$$X_{Tot} = [X \ f_1(X) \ f_2(X) \ f_3(X) \ldots]$$

it therefore also follows that the covariance matrix for any of the individual constituent matrices within $X_{Tot}$, i.e. $X^T X$, $f_1(X)^T f_1(X)$, $f_2(X)^T f_2(X)$, $f_3(X)^T f_3(X)\ldots$, can be retroactively retrieved from the matrix multiplication $X_{Tot}^T X_{Tot}$. After all, horizontally concatenating the different preprocessed versions of the data is effectively a widening of X by adding additional columns to it. In fact, the matrix multiplication $X_{Tot}^T X_{Tot}$ will result in a patchwork of all the possible covariance matrix combinations that can be generated from all the concatenated matrices within $X_{Tot}$. Figure 3 shows the resulting $X_{Tot}^T X_{Tot}$ matrix calculated for the first dataset in the study (a description of the dataset is given in section 3.1) using the preprocessing techniques: $D^1 X$, $D^2 X$, MSC(X), MSC($D^1 X$) and MSC($D^2 X$) concatenated together when forming $X_{Tot}$. The notation used to abbreviate preprocessing techniques is clarified in
As indicated by the arrows in Figure 3, the pure covariance matrices for all of the differently preprocessed versions of $X$ can be found as intact submatrices along the diagonal of $X^T_{Tot}X_{Tot}$. Because of this emerging structure, a single $X^T_{Tot}X_{Tot}$ matrix can be used in combination with the indexing technique shown in Figure 2 to calibrate PLS models containing any combinatorial feature subset of any of the involved preprocessing techniques—or indeed even models containing a mixture of features preprocessed with different methods. As a consequence of this, multiple subpopulations can be positioned within a larger GA-PLS population and be efficiently calibrated in parallel with each other since the data used during PLS can be shared between all chromosomes, regardless of spectral pretreatment method. In order to extract a feature selection corresponding to a specific preprocessing technique from the large $X^T_{Tot}X_{Tot}$ matrix, a small alteration in the chromosome encoding is required. For the genes of a chromosome of length $n$ to map onto the $i^{th}$ $n \times n$ covariance matrix along the diagonal of $X^T_{Tot}X_{Tot}$, the chromosome has to be offset by $(i - 1) \times n$ elements, where $n$ denotes the length of the spectra in $X$. Equivalently, one could also pad the chromosome with zeros in all locations corresponding to features that are outside of the preprocessing-specific region the chromosome should operate in. The concept of padding chromosomes with zeros in order to form preprocessing-specific subpopulations is illustrated in Figure 4.

During the selection, mutation and crossover stages of the GA-PLS, one has the choice of either restricting the subpopulations to evolve exclusively within a preprocessing-specific region, or to allow the individuals to combine with each other free of restrictions across the entire solution space to form complex feature selections containing a mixture of preprocessing methods. When only restricted, preprocessing-specific, subpopulations are used, only the submatrices...
along the diagonal of $X_{\text{Tot}}^T X_{\text{Tot}}$ will be referenced during the kernel PLS. In such cases, all off-diagonal submatrices in $X_{\text{Tot}}^T X_{\text{Tot}}$ do not need to be calculated. When including subpopulations permitted to evolve feature subsets containing variables from a mixture of preprocessing techniques however, all regions of $X_{\text{Tot}}^T X_{\text{Tot}}$ will potentially be referenced.
By including or excluding observations (rows) of \( X_{\text{Tot}} \) according to a desired cross-validation scheme, several different versions of \( X_{\text{Tot}}^T X_{\text{Tot}} \) can be precomputed and stored in a three-dimensional cache, as can be seen illustrated in Figure 5, prior to commencing the GA-PLS. During the RMSE\(_{\text{cv}}\) evaluation stage of the GA-PLS, different versions of \( X_{\text{Tot}}^T X_{\text{Tot}} \) can then be retrieved from the cache and used for calibrating and cross-validating the individuals of a population. This eliminates the need of performing any costly covariance calculations at all within the GA-PLS generational loop.

The full set of steps required to implement the GA-PLS technique described in this section is given as pseudocode in Algorithm 1. In the case where subpopulations are restricted to be confined within their own preprocessing-specific region of \( X_{\text{Tot}}^T X_{\text{Tot}} \), the selection and mutation stages of the GA-PLS are carried out in serial, whilst the RMSE\(_{\text{cv}}\) evaluations can be performed simultaneously in parallel for all subpopulations by sharing a common \( X_{\text{Tot}}^T X_{\text{Tot}} \).

**Algorithm 1** Fast GA-PLS for multiple simultaneous preprocessing methods

1: Preprocess \( X \) with \( p \) different methods and horizontally concatenate into \( X_{\text{Tot}} \)
2: Iteratively include/exclude rows of \( X_{\text{Tot}} \) and \( y \) according to a cross-validation scheme and calculate caches of \( c \) different \( X_{\text{Tot}}^T X_{\text{Tot}} \) and \( X_{\text{Tot}}^T y \) matrices
3: Initialize a population containing \( p \) preprocessing-specific Boolean subpopulations
4: Sample \( k \) \( X_{\text{Tot}}^T X_{\text{Tot}} \) and \( X_{\text{Tot}}^T y \) matrices from the caches and use to batch-evaluate the RMSE\(_{\text{cv}}\) of the entire population using kernel PLS with the indexing technique from section 2.2. Convert RMSE\(_{\text{cv}}\) into fitness
5: for \( \text{gen} \leftarrow 1 \) to \( \text{maxgen} \) do
6:   for \( s \leftarrow 1 \) to \( p \) do
7:     Select parents in subpopulation \#\( s \)
8:     Generate offspring using selected parents
9:     Merge the offspring with subpopulation \#\( s \)
10:    Mutate subpopulation \#\( s \)
11:   end for
12: Sample \( k \) \( X_{\text{Tot}}^T X_{\text{Tot}} \) and \( X_{\text{Tot}}^T y \) matrices from the caches and use to batch-evaluate the RMSE\(_{\text{cv}}\) of the entire population using kernel PLS with the indexing technique from section 2.2. Convert RMSE\(_{\text{cv}}\) into fitness
13: for \( s \leftarrow 1 \) to \( p \) do
14:   Select survivors for subpopulation \#\( s \)
Algorithm 1. Pseudocode of steps needed to efficiently perform GA-PLS on multiple preprocessed version of a dataset. Input data is assumed to be a design matrix $X$, a response vector $y$, a scalar parameter $p$ corresponding to the number of preprocessing methods included in the algorithm, a scalar parameter $c$ indicating the cache size to use, a scalar parameter $k$ ($k \leq c$) describing the number of folds to use during $k$-fold cross-validation and a scalar parameter maxgen controlling the number of generations to run the GA-PLS.

3 | EXPERIMENTAL

In order to experimentally verify the viability of the GA-PLS technique described in section 2.2 (Algorithm 1), an implementation of the algorithm was used to perform simultaneous feature selection and identification of optimal preprocessing technique on two hyperspectral datasets. Hyperspectral imaging data is a particularly suitable type of spectroscopic data to use with the kernel PLS based modeling procedure proposed in$^{12}$ since each pixel of the hyperspectral images are treated as a new observation—which results in inherently overdetermined datasets with substantially more rows than columns. To evaluate the computational efficiency of the proposed GA-PLS method, a benchmark was conducted where the time required for performing GA-PLS on one of the hyperspectral datasets using Algorithm 1 was compared to GA-PLS implementations utilizing more well-known PLS fitting procedures.

3.1 | Description of the two datasets used

1. The first hyperspectral dataset contains absorbance spectra from 40 $50 \times 50 \times 10$ mm sized wood samples of the species Norway spruce ($Pincta abies$) treated with various concentrations of a phosphorous-based flame retardant compound. 35 of these samples can be seen depicted in Supplementary figure 1 a). The hyperspectral images were acquired using a push broom camera (Specim, Oulu, Finland) with 256 bands in the 929-2531 nm wavelength range. The average absorbance spectrum of each of the 40 samples in the dataset is shown in Supplementary figure 2. After the signal acquisition, a 0.4-0.5 mm layer was removed from the surface of each wood sample and sent to an inductively coupled plasma (ICP) analysis where the average phosphorus content of each sample was determined. The laboratory established average phosphorus content (g · kg$^{-1}$) associated with each wood sample is the response variable $y$ of the dataset. The size of the design matrix $X$ containing all the absorbance data is $105 \times 256$. For a more comprehensive description of this dataset the reader is referred to.$^{17}$

2. The second dataset consists of six hyperspectral time series sequences. Each sequence contains 150 images depicting a separate $280 \times 100 \times 18$ mm wood sample of the species Scots pine ($Pinus sylvestris$); resulting in a dataset of $6 \times 150$ images. The six samples can be seen depicted in Supplementary figure 1 b). Initially, all six wood samples were submerged under water and left to soak for 24 hours. When taken from the water after the soaking period the samples were placed on a digital scale, which in turn was positioned underneath a hyperspectral push broom camera (Specim, Oulu, Finland) measuring 200 bands in the 392-1022 nm wavelength region. During the course of 21 hours, hyperspectral images were repeatedly taken of each sample every eight minute as they dried. The digital scale which the wood samples were positioned on top of simultaneously provided an estimate of the average moisture content of the sample for each of the 150 images taken during the 21 hours. The weight-estimated average moisture content (%) in each wood sample at each time step is the response variable of the dataset. The average absorbance spectrum from each of the $6 \times 150$ images in the dataset is shown in Supplementary figure 3. The size of the design matrix $X$ containing all the absorbance data is $104 \times 200$. A more thorough description of this dataset can be found in.$^{18}$

3.2 | Description of the preprocessing techniques used

Preprocessing is often considered a crucial part of the data exploration/model development phase, particularly in the field of hyperspectral imaging.$^{19}$ There are many ways in which spectroscopic data can be preprocessed, most methods however fall into one out of two categories: derivation methods or scatter correction methods. The purpose of derivation is to remove baseline effects in the spectra and/or to enhance spectral features. Scatter correction methods on the other hand seek to remove scatter effects from the spectra, thereby making samples more comparable to each other. Scatter
correction techniques are considered especially relevant for near-infrared data since the NIR region is susceptible to disturbances caused by light scattering.\(^1\) Combinations of preprocessing techniques belonging to these two categories can also be used together when treating a spectral dataset in order to obtain benefits associated with both categories. In our experiments, first (D\(^1\)), second (D\(^2\)) and third order derivation (D\(^3\)) were used as derivation methods. All spectral derivations were carried out using Savitzky-Golay derivation\(^20\) with a window size of 11 and a polynomial order of 3. Multiplicative Scatter Correction (MSC)\(^21\) and Standard Normal Variate (SNV)\(^22\) were used as scatter correction methods. With the restriction that derivation is carried out prior to scatter correction, six additional scatter corrected derivation methods were formed and used in the experiments by combining the above-mentioned techniques, namely MSC(D\(^1\)), SNV(D\(^1\)), MSC(D\(^2\)), SNV(D\(^2\)), MSC(D\(^3\)) and SNV(D\(^3\)). All of the 12 preprocessing techniques used in the experiments are summarized in Table 1.

### 3.3 Algorithm settings

In order to perform feature selection on all the 12 preprocessing techniques listed in Table 1 with a single GA-PLS, a population containing 12 preprocessing-specific subpopulations was used. All elements outside of the relevant region for each subpopulation was padded with zeros as conceptually illustrated in row 1-6 of Figure 4. In addition to the 12 preprocessing-specific subpopulations, a 13\(^{th}\) subpopulation was also included in the population, which was not associated with any particular preprocessing technique but instead was permitted to operate unrestrictedly across the whole search space (as illustrated in the chromosomes of row 7-9 in Figure 4). Each subpopulation contained 512 chromosomes, resulting in a total population size of 512 \(\times\) 13. The number of generated offspring in each generation was set to 512 \(\times\) 13, such that 1024 \(\times\) 13 feature selections were evaluated for each generation of the algorithm.

The fitness of each chromosome was defined as its minimum RMSE\(_{cv}\), across the number of considered latent variables, multiplied with minus one such that a high fitness corresponds to a low RMSE\(_{cv}\) and a low fitness corresponds to a high RMSE\(_{cv}\). All relevant GA-PLS settings used in our experiments are summarized in Table 2. Because parts of GA-PLS algorithms are stochastic and different runs produce different outcomes, the feature selection was repeated ten times for both datasets to increase the robustness of the results. The results from these ten runs were then averaged.

### 3.4 Benchmark of the computational performance of the suggested GA-PLS

The computational efficiency of the GA-PLS technique introduced in section 2.2 (Algorithm 1) was compared against GA-PLS implementations containing four well-established PLS fitting procedures: NIPALS,\(^23\) SIMPLS,\(^24\) Bidiag\(^25\) and modified kernel algorithm #2 (without indexing technique from section 2.2).\(^15\) Dataset 1 (spruce treated with a flame retardant) was used in the benchmark together with the 12 preprocessing techniques listed in Table 1. Each algorithm was used to perform feature selection on the 12 differently preprocessed versions of the dataset and the runtime of

| Abbreviation | Category          |
|--------------|-------------------|
| X            | No preprocessing  |
| D\(^1\)X     | Derivative        |
| D\(^2\)X     | Derivative        |
| D\(^3\)X     | Derivative        |
| MSC(X)       | Scatter correction|
| SNV(X)       | Scatter correction|
| MSC(D\(^1\)X)| Combination      |
| SNV(D\(^1\)X)| Combination      |
| MSC(D\(^2\)X)| Combination      |
| SNV(D\(^2\)X)| Combination      |
| MSC(D\(^3\)X)| Combination      |
| SNV(D\(^3\)X)| Combination      |
each algorithm was measured. For simplicity, the 13th unrestricted subpopulation mentioned in section 3.3 was not included in the benchmark. To measure the effect in calculation time of an increasing population size, the benchmark was performed using subpopulation sizes containing 16, 32, 64, 128, 256, 512 and 1024 individuals. The number of parents and the number of offspring generated were set to the same size as the subpopulation. All benchmarks were performed using MATLAB (R2018a). MATLAB’s built-in plsregress function was used to fit regression coefficients when benchmarking the SIMPLS based GA-PLS. The NIPALS, Bidiag2 and modified kernel algorithm #2 based GA-PLS implementations were written in MATLAB with \(15,23,25\) respectively as reference. The MATLAB code used for estimating regression coefficients with kernel PLS and the indexing technique described in section 2.2 can be found in the supplementary material of \(12\). In addition to the CPU implementation of Algorithm 1, a GPU implementation of the algorithm written in CUDA C was also included in the comparison (executed from MATLAB through the MEX interface) which estimated the regression coefficients of all feature subsets within a population in parallel. All computations were performed using 32-bit floating-point numbers. The motivation for using 32-bit floating-point numbers rather than 64-bit is that most consumer-grade GPUs have a substantially higher throughput when performing arithmetic operations on 32-bit numbers \(26\). So much so that it can be considered wasteful from a hardware utilization perspective to perform calculations in 64-bit precision when the use of 32-bit numbers is numerically acceptable.

In the benchmark, the algorithm settings were chosen according to Table 2. However, because of the tremendous time required to calculate the regression coefficients using NIPALS, SIMPLS, Bidiag2 and modified kernel algorithm #2 at large population sizes, the GA-PLS’s in the benchmark using these PLS methods only ran a single generational cycle, instead of the 200 suggested by Table 2. The reason for this is that benchmarking these algorithms on a single CPU using 200 generations would have taken several years. The measured runtime of one generation for these algorithms was scaled by a constant factor in order to approximate the time required for running 200 generations of the algorithms. Hence, the benchmark results presented in this paper for NIPALS, SIMPLS, Bidiag2 and modified kernel algorithm #2 are only an empirical estimate of the time required to perform a full 200 generation GA-PLS, an estimate based on the assumption that all generations take an equal amount of time to complete. Both the CPU and GPU based implementations of Algorithm 1 however, were measured using the full 200 generations.

### 4 RESULTS

#### 4.1 Variable selection results

Table 3 contains a summary of the average number of active variables, average number of optimal PLS components and the average RMSE\(_{cv}\) of each subpopulation after 200 generations when applying Algorithm 1 to dataset 1. Figure 6 a) shows how the RMSE\(_{cv}\) developed for the individuals of the population over the course of 200 generations. Each chromosome of the population is colored according to its subpopulation belonging. Interestingly, in this dataset none of the
included preprocessing techniques were alone able to surpass the performance of the subpopulation operating on the unpreprocessed absorbance data. The subpopulation operating across the whole search domain unrestrictedly, however, was able to find feature selections which resulted in a slightly lower RMSE\textsubscript{cv} than that of those using regular absorbance data. The unrestricted subpopulation was found to contain active features from a broad range of preprocessing methods after 200 generations. Each included preprocessing method except X and MSC(D\textsuperscript{1}X) was responsible for 8-14\% of the active genes in the average unrestricted chromosome. X and MSC(D\textsuperscript{1}X) were only responsible for roughly 1\% each of the active genes within the unrestricted subpopulation, which can be seen as somewhat surprising considering that the unpreprocessed version of the data X individually performs very well.

### TABLE 3
Summary of results after 200 generations of applying the proposed GA-PLS method (Algorithm 1) to dataset 1 (hyperspectral data of spruce treated with a phosphorous-based flame retardant). Results are averaged across ten trials

| Preprocessing method | Num. var.a | Num. lat.b | RMSE\textsubscript{cv}  |
|----------------------|------------|------------|-------------------------|
| X                    | 44         | 6          | 7394                    |
| D\textsuperscript{1}X | 37         | 5          | 7499                    |
| D\textsuperscript{2}X | 42         | 15         | 7825                    |
| D\textsuperscript{3}X | 85         | 14         | 8247                    |
| MSC(X)               | 38         | 4          | 7514                    |
| SNV(X)               | 38         | 4          | 7473                    |
| MSC(D\textsuperscript{1}X) | 47   | 5          | 8238                    |
| SNV(D\textsuperscript{1}X) | 35   | 2          | 8070                    |
| MSC(D\textsuperscript{2}X) | 45   | 1          | 10687                   |
| SNV(D\textsuperscript{2}X) | 58   | 8          | 8380                    |
| MSC(D\textsuperscript{3}X) | 86   | 14         | 8908                    |
| SNV(D\textsuperscript{3}X) | 92   | 14         | 8687                    |
| Multiple             | 62         | 9          | 6966                    |

aAverage number of active variables in the subpopulations after 200 generations.

bAverage number of optimal latent variables in the subpopulation after 200 generations.

### FIGURE 6
Cross-validated root-mean-squared error, RMSE\textsubscript{cv}, as a function of generation for every member of the population when applying Algorithm 1 to (A) dataset 1 (spruce treated with a phosphorous-based flame retardant) and (B) dataset 2 (drying pine samples). Results are averaged across ten trials.
Table 4 and Figure 6 b) shows the corresponding set of results from applying Algorithm 1 to dataset 2. As seen in Table 4 and Figure 6 b), in this dataset all the included preprocessing techniques managed to improve the $\text{RMSE}_{\text{cv}}$ compared to the unpreprocessed absorbance data. After 200 generations, the unrestricted subpopulation operating on features from a mixture of preprocessing methods again resulted in the lowest $\text{RMSE}_{\text{cv}}$. Although the unrestricted subpopulation did contain approximately twice the number of active variables compared to any of the preprocessing-specific populations, its average optimal number of latent variables was still relatively low. This could indicate that the subpopulations permitted to operate on a mixture of preprocessing methods are evolving a favorable redundancy into their regression models by including multiple features which describe a similar correlation—resulting in models containing many variables which are still characterizable using a low complexity latent space. For dataset 2, the active genes within the unrestricted subpopulation after 200 generations were less uniformly distributed across the different preprocessing methods compared to dataset 1. Moreover, the correlation between the performance of an individual preprocessing method and the extent to which the preprocessing method contributed genes to the unrestricted subpopulation was somewhat more intuitive, i.e. methods performing well on their own in general contributed a larger number of variables to the unrestricted subpopulation. The lowest number of active genes in the unrestricted subpopulation, 0.7 %, originated from the unpreprocessed $\mathbf{X}$, whilst the highest number of active genes, 14 %, originated from $\mathbf{D}^3\mathbf{X}$.

It should be noted that since the algorithm we used chooses the number of PLS components for every generated feature subset simply by selecting the component number yielding the lowest $\text{RMSE}_{\text{cv}}$, the component numbers shown in Table 3 and Table 4 may not be representative of the amount of latent variables a person would have chosen when manually deciding on the number of PLS components. A person manually choosing the number of PLS components, for instance by taking the slope of the $\text{RMSE}_{\text{cv}}$ vs. component number curve into consideration and assessing possible elbow shapes therein, would likely choose more conservative values than some of the values shown in Table 3 and Table 4.

### Benchmark results

Figure 7 shows the results from the benchmark of measuring the runtime of GA-PLS algorithms using different PLS fitting procedures at various subpopulation sizes. Subpopulation sizes whose runtimes were not explicitly measured are linearly interpolated in the figure to provide a clearer view of the trend. As can be seen in the figure, there is a substantial difference in computational efficiency between the GA-PLS implementations using any of the conventional set of PLS fitting procedures—NIPALS, SIMPLS, Bidiag2 and modified kernel algorithm #2—and the GA-PLS

| Preprocessing method | Num. var. $^a$ | Num. lat. $^b$ | $\text{RMSE}_{\text{cv}}$ |
|----------------------|---------------|---------------|-------------------------|
| $\mathbf{X}$         | 25            | 8             | 18.20                   |
| $\mathbf{D}^1\mathbf{X}$ | 24            | 6             | 14.87                   |
| $\mathbf{D}^2\mathbf{X}$ | 23            | 6             | 16.68                   |
| $\mathbf{D}^3\mathbf{X}$ | 24            | 10            | 12.80                   |
| MSC($\mathbf{X}$)    | 19            | 7             | 14.06                   |
| SNV($\mathbf{X}$)    | 20            | 7             | 14.09                   |
| MSC($\mathbf{D}^1\mathbf{X}$) | 23          | 9             | 17.02                   |
| SNV($\mathbf{D}^1\mathbf{X}$) | 24          | 8             | 16.62                   |
| MSC($\mathbf{D}^2\mathbf{X}$) | 22          | 11            | 14.85                   |
| SNV($\mathbf{D}^2\mathbf{X}$) | 22          | 10            | 14.77                   |
| MSC($\mathbf{D}^3\mathbf{X}$) | 23          | 11            | 14.02                   |
| SNV($\mathbf{D}^3\mathbf{X}$) | 24          | 8             | 14.11                   |
| Multiple             | 43            | 7             | 11.42                   |

$^a$Average number of active variables in the subpopulations after 200 generations.

$^b$Average number of optimal latent variables in the subpopulation after 200 generations.
implementation described in section 2.2 (Algorithm 1). Furthermore, the parallel GPU-based version of Algorithm 1 offers an additional massive leap in speedup compared to the CPU version of the same algorithm.

5 | DISCUSSION & CONCLUSIONS

A fast new method for performing GA-PLS subset selection on multiple preprocessed versions of a dataset within a single algorithm was presented. The main benefit of the introduced method is that it enables the performance of multiple spectral pretreatment methods to be explored in parallel with each other in a manner which is substantially less time consuming than previously possible. Indeed, evaluating the GA-PLS performance of multiple pretreatment methods for tall datasets could previously require weeks, months or even years of calculation time; using the technique introduced in this paper the same exploratory search can now be done in an afternoon. The technique we introduced in this paper is primarily intended for strongly overdetermined systems due to two reasons: (1) the method relies on kernel PLS (specifically Modified kernel algorithm #2\textsuperscript{15}), which is susceptible to numerical issues for systems that are not overdetermined. (2) the computational cost savings offered by the covariance indexing technique used in the method increase the more overdetermined the system is.\textsuperscript{12}

The use of feature selections containing variables from a mixture of preprocessing methods was included in this paper primarily to showcase the possibility of doing so and the ease by which it can be done with the introduced method. Although this type of feature selection did result in the lowest RMSE\textsubscript{cv} for both of our evaluated datasets, in many cases it may still not be advisable to use this approach for a few reasons. For instance, it increases the complexity of the resulting model and arguably lowers its interpretability. With the increased combinatorial complexity available to the GA-PLS, the potential of evolving overfitted solutions—which is widely recognized as an ever-present risk when
dealing with GA-PLS—also rises. Additional care should therefore be taken when designing the cross-validation procedure to use in the GA-PLS if such complex models are permitted. In addition to enabling a quick evaluation of the performance of a set of different preprocessing methods, as we have demonstrated here, the introduced method could also be suitable for finding an optimal set of parameters to use together with one particular preprocessing method. For instance, if one has decided to use Savitzky-Golay derivation or smoothing on a dataset, the set of preprocessing methods used in Algorithm 1 can easily be changed into a set containing replicates of the same preprocessing method but with different parameter settings, i.e. different window widths and polynomial degrees in the case of Savitzky-Golay filtering. Different subpopulations can then be appointed to different parameter configurations of the preprocessing method to provide an indication of which parameter settings are best suited for the dataset. Lastly, it should be noted that because kernel PLS is not the most numerically stable PLS procedure, it could be beneficial to use the technique described in this paper as a means of quickly exploring a spectral dataset and how it responds to various pretreatment methods. Then, once a satisfying combination of preprocessing technique and feature subset has been identified, the final PLS model can be recalibrated using a numerically more precise fitting procedure, such as Bidiag2.

ORCID

Petter Stefansson https://orcid.org/0000-0002-6079-6026
Kristian H. Liland https://orcid.org/0000-0001-6468-9423
Ingunn Burud https://orcid.org/0000-0003-0637-4073

REFERENCES

1. Rinnan Å, Van Den Berg F, Engelsen SB. Review of the most common pre-processing techniques for near-infrared spectra. TrAC Trends in Analytical Chemistry. 2009;28(10):1201-1222.
2. Huan ZH, Ke-Wei HU, Xiao-Guang SH, et al. A Variable Selection Method of Near Infrared Spectroscopy Based on Automatic Weighting Variable Combination Population Analysis. Chinese Journal of Analytical Chemistry. 2018;46(1):136-142.
3. Mehmoond T, Liland KH, Snipen L, Sæbo S. A review of variable selection methods in Partial Least Squares Regression. Chemometrics and Intelligent Laboratory Systems. 2012;118:62-69.
4. Saeyes Y, Inza I, Larranaga P. A review of feature selection techniques in bioinformatics. Bioinformatics. 2007;23(19):2507-2517.
5. Nørgaard L, Saudland A, Wagner J, Nielsen JP, Munck L, Engelsen SB. Interval Partial Least-Squares Regression (iPLS): A Comparative Chemometric Study with an Example from Near-Infrared Spectroscopy. Applied Spectroscopy. 2000;54(3):413-419.
6. Song K, Li L, Tesesco LP, et al. Hyperspectral determination of eutrophication for a water supply source via genetic algorithm–partial least squares (GA–PLS) modeling. Science of The Total Environment. 2012;426:220-232.
7. Leardi R, Lupiáñez González A. Genetic algorithms applied to feature selection in PLS regression: how and when to use them. Chemometrics and Intelligent Laboratory Systems. 1998;41(2):195-207.
8. Jie D, Xie L, Fu X, Rao X, Ying Y. Variable selection for partial least squares analysis of soluble solids content in watermelon using near-infrared diffuse transmission technique. Journal of Food Engineering. 2013;118(4):387-392.
9. Durand A, Devos O, Ruckebusch C, Huvenne J. Genetic algorithm optimisation combined with partial least squares regression and mutual information variable selection procedures in near-infrared quantitative analysis of cotton–viscose textiles. Analytica Chimica Acta. 2007;595(1-2):72-79.
10. Reynés C, Souza SD, Sabatier R, Figueres G, Vidal B. Selection of discriminant wavelength intervals in NIR spectrometry with genetic algorithms. Journal of Chemometrics. 2006;20(3-4):136-145.
11. Zhang YA, Ma Q, Sakamoto M, Furutani H. Effects of population size on the performance of genetic algorithms and the role of crossover. Artificial Life and Robotics. 2010;15(2):239-243.
12. Stefansson P, Indahl UG, Liland KH, Burud I. Orders of magnitude speed increase in partial least squares feature selection with new simple indexing technique for very tall data sets. Journal of Chemometrics. 2019:e3141. https://doi.org/10.1002/cem.3141
13. Eiben AE, Smith JE. Introduction to evolutionary computing. Berlin: Springer Berlin Heidelberg; 2015.
14. Leardi R. Genetic algorithm-PLS as a tool for wavelength selection in spectral data sets. In: Data Handling in Science and Technology; 2003:169-196. https://doi.org/10.1016/s0922-3487(03)23006-9
15. Dayal BS, MacGregor JF. Improved PLS algorithms. Journal of Chemometrics. 1997;11(1):73-85.
16. Lindgren F, Geladi P, Wold S. The kernel algorithm for PLS. Journal of Chemometrics. 1993;7(1):45-59.
17. Stefansson P, Burud I, Thiis TK, Gobakken L, Larnøy E. Estimation of phosphorus-based flame retardant in wood by hyperspectral imaging—a new method. Journal of Spectral Imaging. 2018;7:1-9
18. Stefansson P, Fortuna J, Rahmati H, Burud I, Konevskikh T, Martens H. Hyperspectral time series analysis: hyperspectral image data streams interpreted by modeling known and unknown variations. In Data Handling in Science and Technology 2020 Jan 1 (Vol. 32, pp. 305-331). Amsterdam, Netherlands: Elsevier; 2018. https://doi.org/10.1016/B978-0-444-63977-6.00014-6

19. Vidal M, Amigo JM. Pre-processing of hyperspectral images. Essential steps before image analysis. Chemometrics and Intelligent Laboratory Systems. 2012;117:138-148.

20. Savitzky A, Golay MJ. Smoothing and Differentiation of Data by Simplified Least Squares Procedures. Analytical Chemistry. 1964;36 (8):1627-1639.

21. Geladi P, MacDougall D, Martens H. Linearization and Scatter-Correction for Near-Infrared Reflectance Spectra of Meat. Applied Spectroscopy. 1985;3(39):491-500.

22. Barnes RJ, Dhanoa MS, Lister SJ. Standard Normal Variate Transformation and De-Trending of Near-Infrared Diffuse Reflectance Spectra. Applied Spectroscopy. 1989;43(5):772-777.

23. Massart DL. Handbook of Chemometrics and Qualimetrics. Burlington: Elsevier; 1998:336.

24. de Jong S. SIMPLS: An alternative approach to partial least squares regression. Chemometrics and Intelligent Laboratory Systems. 1993;18 (3):251-263.

25. Björck Å, Indahl UG. Fast and stable partial least squares modelling: A benchmark study with theoretical comments. Journal of Chemometrics. 2017;31(8):e2898.

26. Ploskas N, Samaras N. GPU programming in MATLAB. Cambridge, MA: Elsevier; 2016.

SUPPORTING INFORMATION
Additional supporting information may be found online in the Supporting Information section at the end of the article.

How to cite this article: Stefansson P, Liland KH, Thiis T, Burud I. Fast method for GA-PLS with simultaneous feature selection and identification of optimal preprocessing technique for datasets with many observations. Journal of Chemometrics. 2020;34:e3195. https://doi.org/10.1002/cem.3195