



Classifying and analyzing small-angle scattering data using weighted k -nearest neighbors machine learning techniques

Richard K. Archibald,^{a*} Mathieu Doucet,^b Travis Johnston,^a Steven R. Young,^a Erika Yang^a and William T. Heller^b

Received 9 September 2019

Accepted 15 January 2020

Edited by D. I. Svergun, European Molecular Biology Laboratory, Hamburg, Germany

Keywords: small-angle scattering data; machine learning; modeling; *SasView*.

Supporting information: this article has supporting information at journals.iucr.org/j

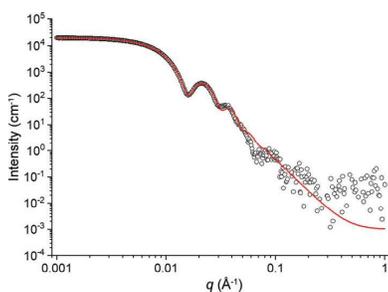
^aComputer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA, and

^bNeutron Scattering Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA. *Correspondence e-mail: archibaldrk@ornl.gov

A consistent challenge for both new and expert practitioners of small-angle scattering (SAS) lies in determining how to analyze the data, given the limited information content of said data and the large number of models that can be employed. Machine learning (ML) methods are powerful tools for classifying data that have found diverse applications in many fields of science. Here, ML methods are applied to the problem of classifying SAS data for the most appropriate model to use for data analysis. The approach employed is built around the method of weighted k -nearest neighbors (wKNN), and utilizes a subset of the models implemented in the *SasView* package (<https://www.sasview.org/>) for generating a well defined set of training and testing data. The prediction rate of the wKNN method implemented here using a subset of *SasView* models is reasonably good for many of the models, but has difficulty with others, notably those based on spherical structures. A novel expansion of the wKNN method was also developed, which uses Gaussian processes to produce local surrogate models for the classification, and this significantly improves the classification accuracy. Further, by integrating a stochastic gradient descent method during post-processing, it is possible to leverage the local surrogate model both to classify the SAS data with high accuracy and to predict the structural parameters that best describe the data. The linking of data classification and model fitting has the potential to facilitate the translation of measured data into results for both novice and expert practitioners of SAS.

1. Introduction

Small-angle scattering (SAS) using either X-rays or neutrons (SAXS or SANS, respectively) is a powerful technique for studying the structure of bulk disordered materials. Metals, ceramics, polymers, nanocomposites, colloids, biomaterials and cement are only a few examples of the possible materials that can be examined with the technique. SAS data from such samples generally lack strongly characteristic features that help guide practitioners through the analysis of the data. Most SAS data analysis relies on the wide variety of models that have been implemented in various software packages (Kline, 2006; Breßler *et al.*, 2015; Doucet *et al.*, 2017). The diversity of the available models, and the possibility that multiple models having very different physical bases can fit the data equally well, can make the process of data analysis challenging, particularly for novice users of the technique. The problem is compounded by the data rates of modern SAXS and SANS instruments. A single experiment can produce a large number of data sets that may require considerable user input to analyze if it is not immediately evident which model should be used for analyzing the data from the material being studied.



Machine learning (ML) methods have proven useful for a variety of classification and identification problems (Mueller *et al.*, 2016; Butler *et al.*, 2018), and the identification of the most appropriate model for fitting a SAS data set is a task well suited to ML. The ability to use the various model functions to calculate a diverse set of model intensity profiles as a training set makes it possible to apply any number of forms of ML to the problem. The most logical form of ML to use is one that evaluates to the same criterion as is used during data analysis, namely the distance between the measured and model SAS intensity profiles, which is directly related to the parameter employed during least-squares minimization. ML methods that employ a distance as an objective function are known as *k*-nearest neighbors (KNN) methods (Cunningham & Delany, 2007). Once the appropriate model for fitting the SAS data has been identified, the next logical step is to determine the parameters that give a model profile that best fits the data. The available SAS data-fitting software packages use nonlinear minimization methods, potentially with user-supplied physically realistic ranges for the starting parameters. With a library of model profiles in hand, such as might be calculated from the extensive body of analytical form and structure factors that are available, other options exist.

The statistics community has spent considerable effort in developing emulators that can approximate stochastic solutions in a functional form, based on either randomly sampled points (Higdon *et al.*, 2004; Kennedy & O'Hagan, 2001; Santner *et al.*, 2003) or advanced sampling methods (Dette & Pepelyshev, 2010; Haaland & Qian, 2011; Johnson *et al.*, 1990; Joseph *et al.*, 2014; Montgomery, 2006). Gaussian processes (GPs) are the preferred methods for approximating both the

mean-field and covariance processes from arbitrary pointwise values under the assumption that the values are independently identically distributed $N(0, \sigma^2)$ random variables. For the case where arbitrary pointwise values are exact, without any error, the mean field is equivalent to reproducing kernel Hilbert spaces (Aronszajn, 1950; Scholkopf & Smola, 2001) with the appropriate matching covariance kernel (Rasmussen & Williams, 2005). Kernel-based approximation methods are at the core of the majority of research being done within the greater high-dimensional function approximation community (Fasshauer, 2007). Similarly to most approximation methods, the bulk of the work assumes continuity of the mean field, with only relatively recent work done for discontinuous mean-field approximation through treed GP methods (Gramacy & Lee, 2008) or partitioning of mean fields into continuous regions (Gorodetsky, 2012). These GP approximation methods can be used with the KNN training data to determine the values of the parameters used to calculate a model SAS intensity profile that are closer to the input SAS data than any of the existing members of the training library.

Here, we present a study of the utility of weighted *k*-nearest neighbors (wKNN) and wKNN with GP for classifying SAS data sets for the most appropriate model for fitting the data. The study utilized training and test model intensity profiles generated using the models implemented in *SasView*'s *sasmodels* package (Doucet *et al.*, 2017). The results demonstrate that the application of GP methods to wKNN makes it possible to identify the most appropriate model for fitting SAS data with good accuracy. Further, by applying stochastic gradient descent (SGD) minimization methods to the results of wKNN with GP using the library of parameters that

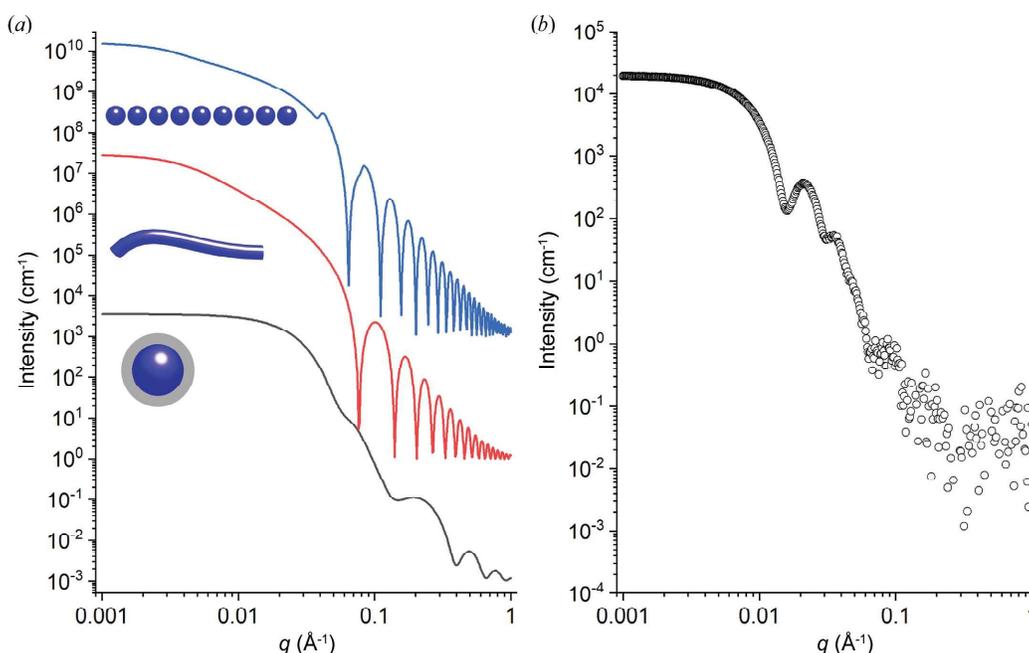


Figure 1 (a) Example model profiles used for training the system, shown with schematics of the structures. The schematics are not shown on the correct relative scale. The black curve was calculated from a core-shell-sphere with a polydisperse core, the red curve was calculated from a flexible cylinder having a polydisperse length and the blue curve was calculated from the linear pearls model having nine pearls. (b) An example core-shell-sphere intensity profile generated with noise for testing the ML methods.

Table 1

Brief descriptions of all the models used in this study, with online links.

All links should be prefixed with <https://www.sasview.org/docs/user/models/> to give their full address.

Model	Brief description	Online documentation link
Adsorbed layer	Scattering from an adsorbed layer on particles	adsorbed_layer.html
Broad peak	Broad Lorentzian-type peak on top of a power-law decay	broad_peak.html
Core multi-shell	This model provides the scattering from a spherical core with one to four concentric shell structures; the SLDs of the core and each shell are specified individually	core_multi_shell.html
Core-shell-sphere	Form factor for a monodisperse spherical particle with a core-shell structure	core_shell_sphere.html
Correlation length	Calculates an empirical functional form for SAS data characterized by a low- Q signal and a high- Q signal	correlation_length.html
DAB	DAB (Debye-Anderson-Brumberger) model	dab.html
Flexible cylinder	Flexible cylinder where the form factor is normalized by the volume of the cylinder	flexible_cylinder.html
Fractal	Calculates the scattering from fractal-like aggregates of spheres following the Texiera reference	fractal.html
Fractal core-shell	Scattering from a fractal structure formed from core-shell-spheres	fractal_core_shell.html
Fuzzy sphere	Scattering from spherical particles with a fuzzy surface	fuzzy_sphere.html
Gauss Lorentz gel	Gauss Lorentz gel model of scattering from a gel structure	gauss_lorentz_gel.html
Guinier	This model fits the Guinier function	guinier.html
Lamellar	Lyotropic lamellar phase with uniform SLD and random distribution	lamellar.html
Lamellar hg	Random lamellar phase with head and tail groups	lamellar_hg.html
Lamellar stack paracrystal	Random lamellar sheet with paracrystal structure factor	lamellar_stack_paracrystal.html
Linear pearls	Linear pearls model of scattering from spherical pearls	linear_pearls.html
Lorentz	Ornstein-Zernicke correlation length model	lorentz.html
Mass fractal	Mass fractal model	mass_fractal.html
Mass surface fractal	Mass surface fractal model	mass_surface_fractal.html
Mono Gauss coil	Scattering from monodisperse polymer coils	mono_gauss_coil.html
Multilayer vesicle	$P(q)$ for a multi-lamellar vesicle	multilayer_vesicle.html
Onion	Onion-shell model with constant, linear or exponential density	onion.html
Peak Lorentz	A Lorentzian peak on a flat background	peak_lorentz.html
Pearl necklace	Colloidal spheres chained together with no preferential orientation	pearl_necklace.html
Poly Gauss coil	Scattering from polydisperse polymer coils	poly_gauss_coil.html
Polymer excl. volume	Polymer excluded volume model	polymer_excl_volume.html
Power law	Simple power law with a flat background	power_law.html
Raspberry	Calculates the form factor, $P(q)$, for a 'raspberry-like' structure	raspberry.html
Sphere	Spheres with uniform scattering length density	sphere.html
Spherical sld	Spherical SLD intensity calculation	spherical_sld.html
Star polymer	Star polymer model with Gaussian statistics	star_polymer.html
Surface fractal	Fractal-like aggregates based on the Mildner reference	surface_fractal.html
Teubner Strey	Teubner-Strey model of microemulsions	teubner_strey.html
Two Lorentzian	This model calculates an empirical functional form for SAS data characterized by two Lorentzian-type functions	two_lorentzian.html
Two-power law	This model calculates an empirical functional form for SAS data characterized by two power laws	two_power_law.html
Unified power R_g	This model employs the empirical multiple level unified exponential/power-law fitting method developed by Beaucage	unified_power_Rg.html
Vesicle	Vesicle model representing a hollow sphere	vesicle.html

produced the training set, it is actually possible to determine the structural parameters required for a given model that best fit the data, thereby providing users of SAS with a new method for data analysis that has considerable potential for automation.

2. Methods

2.1. Training and test data

The SAS data used for training and testing were simulated using the *sasmodels* package that is incorporated in *SasView*. Thirty-nine of the seventy-five model functions implemented in Version 4.1.2 were used for the training and test data. The models that are not included in this study were dropped for being computationally expensive and therefore prohibitive for the demonstration. Herein, the names of the models are used as the class labels, thereby creating a direct relationship with the simulated SAS intensity profile for the ML classifier. The list of the model functions employed is shown in Table 1,

which gives more detail about each model function used and a link to the full description from the online *SasView* documentation. These models correspond to shapes (e.g. star polymer, onion, raspberry) and shape-independent models (e.g. a Gaussian or Lorentzian function).

To train the ML methods, 10 000 samples were generated for each model using physically realistic ranges of parameters supplied to the program using configuration files (see the supporting information, which provides the parameters for the *sasmodels* models employed in the training and testing, along with Python code that validates all ML methods in this paper). In cases where parameters can be polydisperse, such as the radius of a sphere, one of the parameters defining a model was made polydisperse using the methods implemented in *sasmodels*. The model parameters were generated within the specified ranges using Latin hypercube sampling (LHS), which is a standard sampling technique in statistics. LHS uses a stratified sampling technique to create a proper probability distribution of model parameters (McKay *et al.*, 1979).

The set of model profiles, examples of which are shown in Fig. 1(a), and the parameters that produced them were stored for use in the ML methods implemented and in the stochastic gradient descent fitting of the input data, which are described below. The simulated data used to test the ML methods were generated using the same approach as the training data sets, but Gaussian noise was added to the profiles to make them more realistic. An example of a simulated data set for testing the ML methods generated from the ‘core–shell–sphere’ model is shown in Fig. 1(b).

2.2. Weighted k -nearest neighbors

The classic KNN algorithm is a simple ML algorithm used to classify data points into categories, where labeled data are used and classification is defined by the nearest neighbors (Altman, 1992; Cunningham & Delany, 2007). The number of clusters is always defined by the number of different classes in the training data, and for this study $k = 39$. One of the major drawbacks of using regular KNN lies in its majority-rule voting system that makes it vulnerable to skewed distributions of classes, which can result in a higher likelihood of predicting the more common classes. The classic KNN is an unweighted KNN that is based on the Euclidean distance (Altman, 1992). It employs a voting system where, if the top k nearest neighbors are selected, then the class that holds the majority of the k nearest neighbors is selected as the unknown data point’s class.

In the wKNN algorithm (Wettschereck *et al.*, 1997), the weights are assigned on the basis of the distances between points such that the classes with points with shorter distances are given a greater weight than the classes having points with larger distances. Unlike the classical formulation of KNN which calculates the likelihood of predicting the right class on the first try, wKNN uses a local neighborhood to determine classification. By widening the scope of possible classes, the likelihood of finding the correct class is increased. With a larger set of data available, the probability of being able to identify the correct class to which a data set belongs increases, which is broadly true for all ML methods.

Let the unknown SAS data set $I(q)$ be described as a vector $\mathbf{S} = \langle S_1, \dots, S_n \rangle$ that is sampled at n arbitrary points in q , which is the momentum transfer of the scattered X-ray or neutron [$q = (4\pi/\lambda)\sin\theta$, where θ is half the scattering angle and λ is the wavelength of the incident radiation]. The intensity values at each q are the ‘features’ of the profile. We note that the accuracy of this or any other ML method is dependent upon the quality of the training data and the information contained therein. Increasing the q range directly increases the information content of the data. In this study we used the range [10^{-3} \AA^{-1} , 1 \AA^{-1}], which roughly corresponds to the full dynamic range of the SANS instruments at Oak Ridge National Laboratory (Heller *et al.*, 2018).

Let the set of classes of models that are in the training set be denoted \mathbb{C} and let a member w of this set of classes be c_w . Each c_w consists of i member SAS data sets $\mathbf{T}_{c_w,i} = \langle T_{c_w,i,1}, \dots, T_{c_w,i,n} \rangle$ against which an unknown \mathbf{S} is to be clas-

sified. Here, $\mathbf{T}_{c_w,i}$ are sampled at the same n points in q as the unknown \mathbf{S} . We note that, if the unknown data are not sampled at the same points as the training data, this problem can be overcome with simple interpolation of the unknown data. The weighting employed here is the distance d between \mathbf{S} and one member of the class $\mathbf{T}_{c_w,i}$, shown in equation (1):

$$d(\mathbf{S}, \mathbf{T}_{c_w,i}) = \left[\sum_{j=1}^n (\log S_j - \log T_{c_w,i,j})^2 \right]^{1/2}. \quad (1)$$

The large range of intensities in a single profile, as can be seen in Fig. 1, is addressed by working with the logarithm of the intensity. It is important to note that both training and testing data are assumed to be calibrated. It must also be noted that, if calibration of data is not done as a pre-processing step before classification, then the distance function of the classifier, given in equation (1), can be substituted with a similarity measure that is not sensitive to calibration. Alternatively, the unknown data could be preprocessed by applying a scale factor, such as the ratio of the integrated intensities of the model and unknown profiles, to the unknown data. Doing so would increase the computational cost of the method and decrease the accuracy of the prediction.

The kernel function for the wKNN classifier, $K(d(\mathbf{S}, \mathbf{T}_{c_w,i}))$, is defined according to equation (2):

$$K(d(\mathbf{S}, \mathbf{T}_{c_w,i})) = \frac{1}{d(\mathbf{S}, \mathbf{T}_{c_w,i})}. \quad (2)$$

Then, the probability that \mathbf{S} is a member of the class c_w is given by equation (3):

$$p(c_w|\mathbf{S}) = \frac{\sum_{i \in c_w} K(d(\mathbf{S}, \mathbf{T}_{c_w,i}))}{\sum_{c_w \in \mathbb{C}} \sum_{i \in c_w} K(d(\mathbf{S}, \mathbf{T}_{c_w,i}))}. \quad (3)$$

The class c_w to which \mathbf{S} is estimated to belong (*i.e.* what \mathbf{S} is classified to be) is denoted $M(\mathbf{S})$ and is given by equation (4):

$$M(\mathbf{S}) = \max_{c_w \in \mathbb{C}} p(c_w|\mathbf{S}). \quad (4)$$

2.3. Gaussian process kernel formulation

We briefly introduce the noise-free Gaussian process using the kernel formulation given by Rasmussen & Williams (2005) and adapt it to our notation. A GP is defined as a collection of random variables such that any finite subset of the collection follows a Gaussian distribution. Suppose we are given a set \mathcal{S}_N of unique data points, $\mathcal{S}_N = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, in a bounded domain $\mathcal{S}_N \subset \mathbb{R}^d$. For a smooth function f , we define the vector

$$f_{\mathcal{S}_N} = \begin{pmatrix} f(\mathbf{x}_1) \\ \vdots \\ f(\mathbf{x}_N) \end{pmatrix}. \quad (5)$$

The covariance matrix for the set of points \mathcal{S}_N and $\tilde{\mathcal{S}}_N$ is defined as

$$\mathbf{K}(\mathcal{S}_N, \tilde{\mathcal{S}}_N) = \begin{pmatrix} k(\mathbf{x}_1, \tilde{\mathbf{x}}_1) & \cdots & k(\mathbf{x}_1, \tilde{\mathbf{x}}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \tilde{\mathbf{x}}_1) & \cdots & k(\mathbf{x}_N, \tilde{\mathbf{x}}_N) \end{pmatrix}, \quad (6)$$

where $k(\mathbf{x}_i, \tilde{\mathbf{x}}_j)$ is the covariance function or kernel for $f_{\mathcal{S}_N}$. In this study we use the commonly used stationary non-degenerate squared exponential kernel,

$$k(\mathbf{x}_i, \tilde{\mathbf{x}}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \tilde{\mathbf{x}}_j\|^2}{2l^2}\right), \quad (7)$$

where l is a length parameter for weighting the distance between points. Details about other valid kernels are given by Rasmussen & Williams (2005), along with a more detailed explanation of the GP method.

Then, we use the standard random variable notation, $F(\mathbf{x}_i)$ to denote the GP at the point \mathbf{x}_i . The $f(\mathbf{x}_i)$ are the realization of the GP at the point \mathbf{x}_i . Here, we assume that there exists additive, independent and identically distributed Gaussian noise with variance σ^2 . Then, the posterior distribution of $F(\mathbf{x})$ for an arbitrary point $\mathbf{x} \in \mathbb{R}^d$, given the data \mathcal{S}_N and $f_{\mathcal{S}_N}$, is

$$F(\mathbf{x})|\mathcal{S}_N, f_{\mathcal{S}_N} \simeq \mathcal{N}\left(\mathbf{K}(\mathbf{x}, \mathcal{S}_N) [\mathbf{K}(\mathcal{S}_N, \mathcal{S}_N) + \sigma^2 \mathbf{I}]^{-1} f_{\mathcal{S}_N}, \mathbf{K}(\mathbf{x}, \mathbf{x}) - \mathbf{K}(\mathbf{x}, \mathcal{S}_N) [\mathbf{K}(\mathcal{S}_N, \mathcal{S}_N) + \sigma^2 \mathbf{I}]^{-1} f_{\mathcal{S}_N}, \mathbf{K}(\mathcal{S}_N, \mathbf{x})\right). \quad (8)$$

Here, \mathbf{I} is the identity matrix. In functional form, the mean GP function is given by

$$\overline{f(\mathbf{x})} = \mathbf{K}(\mathbf{x}, \mathcal{S}_N) [\mathbf{K}(\mathcal{S}_N, \mathcal{S}_N) + \sigma^2 \mathbf{I}]^{-1} f_{\mathcal{S}_N}, \quad (9)$$

with variance

$$\begin{aligned} \mathbb{V}[\overline{f(\mathbf{x})}] \\ = \mathbf{K}(\mathbf{x}, \mathbf{x}) - \mathbf{K}(\mathbf{x}, \mathcal{S}_N) [\mathbf{K}(\mathcal{S}_N, \mathcal{S}_N) + \sigma^2 \mathbf{I}]^{-1} f_{\mathcal{S}_N}, \mathbf{K}(\mathcal{S}_N, \mathbf{x}). \end{aligned} \quad (10)$$

An implementation of GP for wKNN, referred to as gpKNN, was developed. The way that the GP improves wKNN is shown schematically in Fig. 2. Surrogate models are produced for each class using the GP and the training data available for each class. These GP surrogates produce a global approximation for each class, which often leads to finding fits to the classification data that are better than existing members of the set of training data sets. By leveraging the continuity of the space of the curves that result from the ranges of model parameters sampled, the accuracies of the predictions produced by gpKNN are generally better than those produced by wKNN.

2.4. Evaluating wKNN and gpKNN performance

In order to gauge how accurate the wKNN and gpKNN classifications are at identifying the correct model to use for data analysis, a set of unknowns encompassing all implemented classes was generated and all $p(c_w|\mathbf{S})$ were calculated [equation (3)]. The percentage of tests where $p(c_w|\mathbf{S})$ correctly identifies the class was determined, as were the percentages of tests where the correct class was one of the top two or top three $p(c_w|\mathbf{S})$ values. When used with the stochastic gradient descent fitting, described below, the parameters associated with the best models in the various identified sets of training

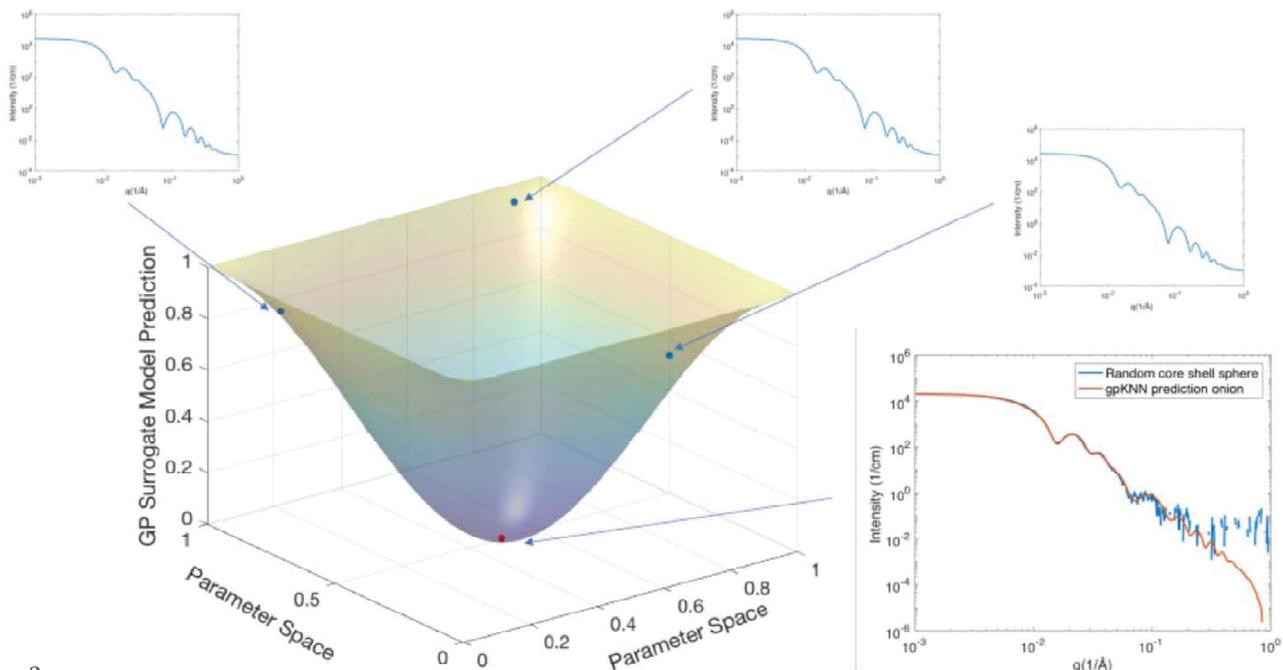


Figure 2

The neighborhood of the closest model found in a class is used to generate interpolations of class members to determine whether a better fit to the data can be found from within that class. We have reduced the parameter space for visualization purposes.

Table 2

Performance of the wKNN method for identifying the correct model that should be used for fitting a SANS data set.

The ordering of the models in the table is based on the ‘Top 3’ column. Subsequent orderings are based on the ‘Top 2’ and ‘Top’ columns.

Model	Top	Top 2	Top 3	Model	Top	Top 2	Top 3	Model	Top	Top 2	Top 3
Lorentz	100	100	100	Mass fractal	94	97	99	Raspberry	79	92	95
Star polymer	100	100	100	Multilayer vesicle	88	97	99	Fractal core-shell	61	88	94
Guinier	100	100	100	Mono Gauss coil	61	97	99	Gauss Lorentz gel	56	76	94
DAB	100	100	100	Broad peak	90	96	99	Teubner Strey	85	90	92
Mass surface fractal	98	100	100	Pearl necklace	83	95	99	Spherical sld	70	82	92
Two-power law	98	100	100	Unified power R_g	86	94	99	Lamellar	56	79	90
Linear pearls	90	100	100	Adsorbed layer	87	92	99	Flexible cylinder	71	78	88
Peak Lorentz	99	99	100	Power law	97	97	98	Fractal	52	68	82
Correlation length	98	99	100	Lamellar stack paracrystal	92	97	98	Core multi-shell	53	68	76
Polymer excl. volume	93	99	100	Two Lorentzian	92	96	98	Sphere	9	26	70
Surface fractal	97	98	100	Lamellar hg	72	88	97	Onion	22	38	52
Poly Gauss coil	77	98	100	Vesicle	73	84	97	Fuzzy sphere	7	14	33
Gaussian peak	97	98	99	Gel fit	72	90	96	Core-shell-sphere	0	0	1

model profiles were employed as the starting conditions for the fitting.

2.5. Stochastic gradient descent data fitting

Stochastic gradient descent (SGD) is a popular method for function optimization in ML because it is very effective for large-scale optimization problems (Bottou & Bousquet, 2008). When the dimension of the optimization parameter is high, as is the case in large-scale ML problems, a traditional gradient descent optimization becomes computationally prohibitive, or even intractable, due to memory and processing limitations. In contrast, SGD accelerates the iterative optimization step by reducing the number of data points used to calculate the gradient, potentially down to a single data point. The SGD for parameters θ and an objective function $J(\theta)$ at a given data point pair (x_i, y_i) considered at that step of the SGD is

$$\theta = \theta - \alpha \nabla_{\theta} J(\theta, (x_i, y_i)). \quad (11)$$

Here, α is the step size of the iterative optimization process. In the present work, SGD is used to optimize the fit of a model profile to the input data set. The parameters θ are the various fitting parameters in the models implemented in *SasView*. The parameters for the top model identified by gpKNN served as the starting point for the optimization. The quality of the fit to the data is evaluated using the distance between the model intensity profile and the input data shown in equation (1). The coupling of gpKNN to SGD is referred to as sgdKNN and its performance in identifying the correct model class was evaluated using the same approach as for wKNN and gpKNN.

3. Results

An example prediction by wKNN for the ‘core-shell-sphere’ data set shown in Fig. 1(b) is presented in Fig. 3. The best model class for the input data was identified as the ‘onion’ model, which is a multi-shell spherical model that is closely related to the true class, and the profile from the onion training set is presented with the data in Fig. 3. The closest core-shell training data set is also presented with the data in Fig. 3, and is only the fourth most appropriate model

according to the wKNN classifier. This particular training set is clearly less similar to the model than the curve taken from the set of onion profiles and demonstrates that the accuracy of wKNN is bound by the breadth of the training set.

The overall performance of wKNN was determined for 100 randomly chosen test models from each of the various models in the training set against the training set calculated using *SasView*. The results of the test for the top three predictions are presented in Table 2 as the percentage of time that the right model was found to be in one of the top three selections. In many cases, wKNN correctly identified the most appropriate model as being within the three most appropriate model types. As a summary of the information in the table, the average percentage of time that the correct model was the top choice was 75.8%. Further, 84.9% of the time it was in the top two choices, and the correct model was in the top three choices 90.6% of the time. In light of the fact that the training set consisted of 39 models, the results demonstrate that using ML methods to assist users in identifying how to analyze their data

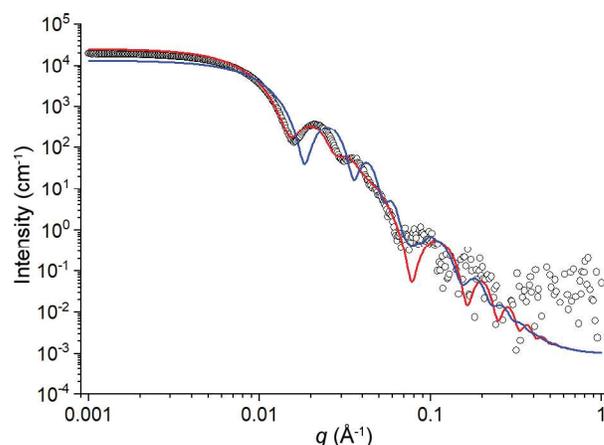


Figure 3 The class identified by wKNN to be nearest to the core-shell-sphere input data shown (circles) in Fig. 1 is the onion class. The onion class training data set (red) is the nearest member of any class to the data. The core-shell-sphere class curve (blue) was the best member of that particular class, but the core-shell-sphere class of models was only the fourth-nearest class of models to the simulated data.

Table 3

Performance of the gpKNN method for identifying the correct model that should be used for fitting a SANS data set.

The ordering of the models in the table is based on the ‘Top 3’ column. Subsequent orderings are based on the ‘Top 2’ and ‘Top’ columns.

Model	Top	Top 2	Top 3	Model	Top	Top 2	Top 3	Model	Top	Top 2	Top 3
Star polymer	100	100	100	Linear pearls	88	96	100	Multilayer vesicle	74	86	98
DAB	100	100	100	Lamellar stack paracrystal	82	95	100	Surface fractal	96	96	97
Two Lorentzian	99	100	100	Gel fit	87	99	99	Mono Gauss coil	87	95	97
Unified power R_g	99	100	100	Flexible cylinder	94	98	99	Spherical sld	64	89	95
Polymer excl. volume	98	100	100	Two-power law	92	98	99	Core multi-shell	76	86	94
Mass surface fractal	98	100	100	Gauss Lorentz gel	77	96	99	Core-shell-sphere	67	86	91
Teubner Strey	97	100	100	Broad peak	72	93	99	Fractal	55	70	84
Guinier	93	100	100	Adsorbed layer	88	93	99	Fuzzy sphere	9	48	83
Lorentz	90	100	100	Pearl necklace	97	98	98	Lamellar hg	45	66	82
Gaussian peak	98	99	100	Mass fractal	86	97	98	Sphere	50	67	77
Correlation length	93	99	100	Raspberry	72	97	98	Vesicle	45	59	76
Peak Lorentz	89	99	100	Power law	91	96	98	Fractal core-shell	44	55	75
Poly Gauss coil	85	97	100	Onion	77	93	98	Lamellar	24	41	69

Table 4

Performance of the sgkKNN method for identifying the correct model that should be used for fitting a SANS data set.

The ordering of models in the table is based on the ‘Top 3’ column. Subsequent orderings are based on the ‘Top 2’ and ‘Top’ columns.

Model	Top	Top 2	Top 3	Model	Top	Top 2	Top 3	Model	Top	Top 2	Top 3
Star polymer	100	100	100	Linear pearls	91	97	100	Multilayer vesicle	76	87	98
DAB	100	100	100	Lamellar stack paracrystal	85	97	100	Surface fractal	97	97	98
Two Lorentzian	99	100	100	Gel fit	87	99	99	Mono Gauss coil	92	97	98
Unified power R_g	100	100	100	Flexible cylinder	95	98	99	Spherical sld	78	91	96
Polymer excl. volume	100	100	100	Two-power law	94	99	99	Core multi-shell	79	91	95
Mass surface fractal	100	100	100	Gauss Lorentz gel	79	97	99	Core-shell-sphere	80	90	94
Teubner Strey	98	100	100	Broad peak	82	94	99	Fuzzy sphere	33	63	86
Guinier	99	100	100	Adsorbed layer	92	94	99	Lamellar hg	50	75	85
Lorentz	98	100	100	Mass fractal	91	98	99	Fractal	67	78	84
Gaussian peak	99	100	100	Onion	84	93	99	Vesicle	64	63	80
Correlation length	96	99	100	Pearl necklace	98	98	98	Sphere	51	76	78
Peak Lorentz	97	99	100	Raspberry	74	97	98	Fractal core-shell	58	59	76
Poly Gauss coil	88	98	100	Power law	93	97	98	Lamellar	42	60	72

has considerable potential. However, there are clear cases where the limitations of wKNN become evident. The core-shell-sphere model is the most extreme example of where the correct model was not within the three highest $p(c_w|\mathbf{S})$, even though it had the fourth highest $p(c_w|\mathbf{S})$. In fact, the five worst performing models in the set share a common feature: they are sphere-based shapes. The curves in these training sets presumably have a basic level of commonality that creates a large number of possibilities for being near to the input data set, such as can be seen in Fig. 3, leading to misidentification of the correct model for the input data. In the example presented in Fig. 3, there was simply a curve in the onion training set that was closer to the test data than any of the curves in the core-shell-sphere training set. If more of the cylinder-based models, of which there are more than ten, had been included in the set of models used to construct training sets, similar results may have been observed in that set of models. Instead, the inclusion of a single cylinder-based model in the set appears to have reduced classification ambiguity arising from a larger population of closely related models.

Table 3 displays the results of testing gpKNN with the same 100 randomly selected test data sets from each class that were used in the evaluation of wKNN. Again, the top three predictions were determined and the percentage of time that

the correct model was predicted to be in one of the top three models was determined. As was true in the wKNN testing, the correct model was predicted to be within the top three predictions over 99% of the time for over half of the classes of test data. However, the use of surrogate models in gpKNN improved the performance of the prediction for all models. Specifically, gpKNN considerably increased the number of times the correct model was in the top three predicted classes. To provide a broad comparison against the wKNN testing, the average percentage of time that the correct model was the top choice of gpKNN was 78.9%. It was among the top two choices 89.4% of the time, and 94.9% of the time it was among the top three choices. The results demonstrate a clear improvement in the overall classification performance by gpKNN. It is important to note that the number of models for which the correct model was the top choice did not improve significantly for the models that were very accurately classified by wKNN, and arguably decreased for many classes, on the basis of the numbers in the table. Only one model was correctly identified as the top choice less than 10% of the time (the ‘fuzzy sphere’ model). The results indicate that the application of the GP introduced a degree of uncertainty in the direct classification, even though the general character of the various classes of models made it possible to place the

correct class within the top three closest classes of models. The improvement in the ability to narrow down the right class of model to use during data analysis to three possibilities when analyzing data from an experiment is of real value when analyzing a large number of data sets where the fit to the data is not known *a priori*.

The results of the quality of predictions that are obtained by coupling SGD with gpKNN (sgdKNN) are presented in Table 4. The same 100 randomly selected test data sets from the various models used to generate the training sets were employed for the performance characterization. Overall, the integration of SGD minimization from the parameter set that created the nearest model in the various models to the input data improved the ability of the classifier to identify the correct model from which the input data were generated. The correct model was identified 84.3% of the time, and the top two and three choices contained the correct model 91.8 and 95.5% of the time, respectively. The source of the improvement can be understood through Fig. 4, which returns to the core-shell-sphere model and the three closer classes originally identified by gpKNN for the test input data presented in Fig. 1. Starting from the sets of parameters for each training set profile, the SGD minimization searches the parameter space for new parameters that reduce the distance between the model profile from the class and the input data. In some cases, such as for the correct class of model, considerable improvement can be obtained. Inappropriate model classes that simply had a training set member near to the input data were only marginally improved. The onion class model was closest to the core-shell-sphere to begin with, but parameters could not be found that fitted the data as well as the simple core-shell-sphere model did. The quality of the fit that the sgdKNN core-shell-sphere model found to the input data can be seen in Fig. 4. In principle, the onion model should be able to fit data that arise from a core-shell-sphere structure, but the additional parameters required to define the structure presumably complicate the minimization search in such a way that the

search becomes trapped in a local minimum. The main downside to sgdKNN is in the greatly increased computational cost of performing the SGD minimization because thousands of profiles are calculated during the minimization. However, the benefits of immediately having the data analysis result provided with the most appropriate class for fitting the input data are considerable and this is the main benefit of using an ML method, such as KNN, to identify a range of likely model choices.

4. Discussion and conclusion

The challenges in determining the most appropriate way to analyze SAS data are often a source of frustration for users of the techniques because of the nature of the data and the many different ways of analyzing them. The present work demonstrates that it is possible to use ML methods to narrow down the choices for fitting SAS data from the diverse set that are available, such as those that are implemented in the *SasView* software package, to a few that could be manually fitted with much less user effort. The ML method that serves as the foundation for the present work, wKNN, employs a distance function directly related to the target functions used in least-squares minimization for SAS data analysis. As a result, the comparison of the various models in the training sets against the input data arguably uses the most direct measurement available among ML methods.

The use of wKNN has other advantages, as well as drawbacks. A key advantage is that by using training sets of data directly, rather than abstractions of feature sets, such as are used in neural networks, it becomes possible to preprocess the unknown data or the training data prior to comparison with equation (1). For example, as was noted in the *Methods* section, it may be necessary to apply a scale factor in cases where the unknown data were not properly calibrated. Similarly, it may be necessary to apply a baseline to the training data to be consistent with the input data. Such operations can

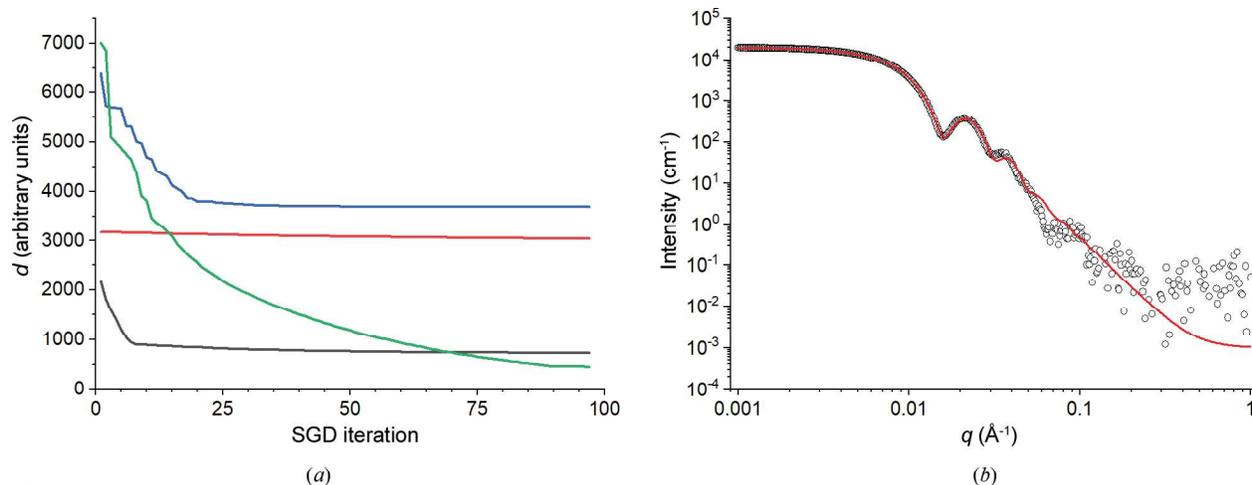


Figure 4 (a) An illustration of the convergence of the SGD minimization for the test core-shell-sphere data shown in Fig. 1. The starting models were the best found by gpKNN for the top three identified classes, namely onion (black), spherical SLD (red), flexible cylinder (blue) and core-shell-sphere (green). (b) The best fitting core-shell-sphere model curve found using sgdKNN (red) for the input core-shell-sphere data set (open circles).

be readily performed when using wKNN but may not be possible when working with neural networks. Importantly, the training data can easily be convoluted with the instrument resolution function (*i.e.* smearing parameters) when performing the classification using any of the wKNN-based approaches presented herein. While doing so is of limited importance when working with SAXS data, particularly from synchrotrons, it is absolutely critical when working with SANS data. The configurability of most SANS instruments makes accounting for the instrument resolution function when working with abstractions of feature sets not only difficult but computationally expensive, because feature set abstractions need to be produced for all reasonable instrument configurations that may be employed. Note that the smearing of features in the data caused by the instrument resolution function of a SANS instrument is not negligible and will affect the prediction of any classification method, including those based on wKNN that are presented here.

Computational expense is arguably one of the greatest drawbacks of wKNN and related methods. All profiles in the training data sets must be compared against each test data set input for classification. As the wKNN database grows, the computational expense of analyzing the data dominates the cost of performing a prediction. Currently, there exist many hardware and software solutions for dealing with massive quantities of data, but addressing the matter is beyond the scope of this project. Regardless, for extremely large and complex classification problems, the desire to minimize the size of the training set would be strong and might have an impact on the power of the classifier. When coupled with SGD, as was done here, the computational cost of performing a single classification increases considerably. However, such methods may make it possible to use a smaller training set than when they are not used. Fortunately, many of the calculations that must be performed are highly amenable to parallelization, such that a moderate-sized computer cluster could improve performance considerably, even when incorporating the instrument resolution function of a SANS instrument.

The method described herein has the greatest potential to benefit the operation of beamlines at user facilities where the breadth of scientific topics studied is the greatest. One could envision having the *sgdKNN* method coupled to the output of the data-reduction process in a streamlined workflow. Shortly after the experiment ends, a clearer path forward towards data analysis and a scientific result would be presented to the user than is often available at present. Mail-in programs at user facilities would also benefit greatly because the methods presented herein could help the staff efficiently collect and analyze data for the mail-in program users of the facility.

Funding information

The following funding is acknowledged: ORNL LDRD – Director’s R&D (grant No. 32112626). Research sponsored by

the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory, managed by UT-Battelle, LLC, for the US Department of Energy (LDRD-8235). This work benefited from the use of the *SasView* application, originally developed under NSF award DMR-0520547. *SasView* contains code developed with funding from the European Union’s Horizon 2020 research and innovation programme under the SINE2020 project, grant agreement No. 654000.

References

- Altman, N. S. (1992). *Am. Stat.* **46**, 175–185.
- Aronszajn, N. (1950). *Trans. Am. Math. Soc.* **68**, 337.
- Bottou, L. & Bousquet, O. (2008). *Adv. Neural Inf. Process. Syst.* **20**, 161–168.
- Breßler, I., Kohlbrecher, J. & Thünemann, A. F. (2015). *J. Appl. Cryst.* **48**, 1587–1598.
- Butler, K. T., Davies, D. W., Cartwright, H., Isayev, O. & Walsh, A. (2018). *Nature*, **559**, 547–555.
- Cunningham, P. & Delany, S. J. (2007). *Multiple Classifier Systems*, **34**, 1–17.
- Detle, H. & Pepelyshev, A. (2010). *Technometrics*, **52**, 421–429.
- Doucet, M., Cho, J. H., Gervaise, A., Bakker, J., Bouwman, W., Butler, P., Campbell, K., Gonzales, M., Heenan, R., Jackson, A., Juhas, P., Kienzle, P., Krzywon, J., Markvardsen, A., Nielsen, T., O’Driscoll, L., Potrzebowski, W., Ferraz Leal, R., Richter, T., Rozycko, P., Snow, T. & Washington, A. (2017). *SasView*. Version 4.1.2. <https://www.sasview.org/>.
- Fasshauer, G. F. (2007). *Meshfree Approximation Methods with MATLAB*. River Edge: World Scientific Publishing Co.
- Gorodetsky, A. A. A. (2012). Master’s thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA.
- Gramacy, R. B. & Lee, H. K. H. (2008). *J. Am. Stat. Assoc.* **103**, 1119–1130.
- Haaland, B. & Qian, P. Z. G. (2011). *Ann. Statist.* **39**, 2974–3002.
- Heller, W. T., Cuneo, M., Debeer-Schmitt, L., Do, C., He, L., Heroux, L., Littrell, K., Pingali, S. V., Qian, S., Stanley, C., Urban, V. S., Wu, B. & Bras, W. (2018). *J. Appl. Cryst.* **51**, 242–248.
- Higdon, D., Kennedy, M., Cavendish, J. C., Cafo, J. A. & Ryne, R. D. (2004). *SIAM J. Sci. Comput.* **26**, 448–466.
- Johnson, M. E., Moore, L. M. & Ylvisaker, D. (1990). *J. Statist. Plann. Inference*, **26**, 131–148.
- Joseph, V. R., Dasgupta, T., Tuo, R. & Wu, C. F. J. (2015). *Technometrics*, **57**, 64–74.
- Kennedy, M. C. & O’Hagan, A. (2001). *J. R. Stat. Soc. B*, **63**, 425–464.
- Kline, S. R. (2006). *J. Appl. Cryst.* **39**, 895–900.
- McKay, M. D., Beckman, R. J. & Conover, W. J. (1979). *Technometrics*, **21**, 239–245.
- Montgomery, D. C. (2006). *Design and Analysis of Experiments*. New York: John Wiley and Sons.
- Mueller, T., Gilad Kusne, A. & Ramprasad, R. (2016). *Rev. Comput. Chem.* **29**, 186–273.
- Rasmussen, C. E. & Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning*. Cambridge: MIT Press.
- Santner, T. J., Williams, B. J. & Notz, W. I. (2003). *The Design and Analysis of Computer Experiments*. New York: Springer.
- Scholkopf, B. & Smola, A. J. (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. Cambridge: MIT Press.
- Wettschereck, D., Aha, D. W. & Mohri, T. (1997). *Artif. Intell. Rev.* **11**, 273–314.