Computers & Geosciences 140 (2020) 104501

Contents lists available at ScienceDirect

Computers and Geosciences

journal homepage: www.elsevier.com/locate/cageo

Research paper

Improved well log classification using semisupervised Gaussian mixture models and a new hyper-parameter selection strategy

Michael W. Dunham *, Alison Malcolm, J. Kim Welford

Department of Earth Sciences, Memorial University of Newfoundland, St. John's, NL, A1B 3X5 Canada

ARTICLE INFO

Keywords:
Lithofacies
Well logs
Semisupervised
Classification
Hyper-parameter selection

ABSTRACT

Well log classification, the process of mapping well log measurements to lithofacies identified from core samples, is a common procedure in the oil and gas industry. Manually assigning lithofacies to the wire-line log measurements without core can be time consuming, and can also introduce a bias. Supervised machine learning algorithms are commonly used to automate this process, but they are prone to overfitting when the training data are scarce, which is common for well log classification problems. Semisupervised machine learning algorithms are designed for classification problems with minimal training data, and we adopt a semisupervised Gaussian mixture model (ssGMM) method to solve this problem. The dataset we consider for our study is from a machine learning competition held in 2016 and we simulate a semisupervised scenario by assuming only one out of the ten wells is the labeled data. We apply ssGMM to this well log dataset and compare its performance to the supervised method that was the winner of this competition, XGBoost. To try and improve the performance of both ssGMM and XGBoost, we also introduce a new hyper-parameter selection strategy that simultaneously uses the mean and standard deviation cross-validation scores, compared to the default procedure that only utilizes the mean cross-validation scores. Our results indicate that ssGMM is able to slightly outperform XGBoost in our semisupervised context, which supports the suggestion that semisupervised algorithms are more appropriate in low training data situations. We also show that our new hyper-parameter selection technique selects hyper-parameters for ssGMM that perform better on the testing data, but the performance is mixed for XGBoost.

1. Introduction

In recent years, there has been an increasing demand for data science across many disciplines for two reasons. One, technological advances have improved data storage capabilities as well as how data can be obtained (e.g., real-time data). Manually interpreting data that are exponentially growing in volume has obvious management and analysis challenges. Machine learning automates pattern recognition of big data in an efficient manner. The second motivation for data science is derived from a shortcoming in inversion. In many scientific problems, we are trying to understand a model that generates the data we observe. For inversion, this relationship between data and model is conventionally expressed as $d = \mathbf{G}m$ where d is the observed dataset, m is the model, and $\mathbf{G}$ is the forward operator that contains the *explicit* physics or mathematics needed to relate the observed data to the model. Simplistically, the model can be described by taking the inverse of the forward operator, $m = \mathbf{G}^{-1}d$. Performing this operation is not trivial if $\mathbf{G}$ is complicated, or non-linear, or perhaps it is not even possible if $\mathbf{G}$ is unknown for the problem of interest. Machine learning algorithms learn an *implicit* mapping to go from d to m without the need for *explicit* relationships to be programmed.

One scenario where an explicit mathematical relationship is unknown for is well log classification. The objective for well log classification is to map well log measurements (e.g., gamma ray, density, neutron porosity, resistivity, sonic, etc.) to lithofacies (commonly identified from core samples). This problem has been widely addressed using both supervised learning (SL) and unsupervised learning (UL). Neural networks were the first supervised machine learning algorithms applied to well log classification (Baldwin et al., 1990; McCormack, 1991; Rogers et al., 1992), and they continued to be popular for the next two decades (Benaouda et al., 1999; Saggaf and Nebrija, 2000; Maiti et al., 2007; Al-Bulushi et al., 2009; Malvić et al., 2010; Al-Bulushi et al., 2012). Other supervised implementations have included k-nearest neighbors (Dubois et al., 2007), support vector machines (Wang et al., 2013; Hall, 2016) and ensemble methods (Bestagini et al., 2017; Keynejad et al., 2019). Bayesian-based techniques have

* Corresponding author.
E-mail addresses: mwdunham@mun.ca (M.W. Dunham), amalcolm@mun.ca (A. Malcolm), kwelford@mun.ca (J.K. Welford).

https://doi.org/10.1016/j.cageo.2020.104501
Received 28 October 2019; Received in revised form 14 April 2020; Accepted 15 April 2020
Available online 22 April 2020

also been popular for facies classification problems, and these methods encompass both supervised and unsupervised learning. A common supervised Bayesian technique is a Naïve Bayes classifier (Li and Anderson-Sprecher, 2006; Dubois et al., 2007; Grana et al., 2017), and Gaussian mixture models (GMMs) are popular for unsupervised implementations (Gallop, 2006; Grana et al., 2017; Hardisty and Wallet, 2017; Wallet and Hardisty, 2019). In recent years, unsupervised GMMs have been extended to include transition probabilities from hidden Markov models (Lindberg and Grana, 2015; Feng et al., 2018a,b).

A challenge with well log classification is the availability of ground truth, labeled data for supervised learning. It is relatively cheap to collect unlabeled data (wireline data), but obtaining ground truth labels for the unlabeled data can be difficult because extracting, preserving, and storing core samples is costly. Therefore, these classification problems commonly have a paucity of labeled data. For instance, Dubois et al. (2006) train a neural network using 14 wells with core and they use that mapping to predict the lithofacies for 1364 wells without core samples. In this situation, the supervised classifier is trained using roughly 1% of the data and this assumes that the classifier can generalize in classifying the remaining 99%. This may be a poor assumption, in general, because it is known that training a supervised classifier on a small training set can lead to overfitting, and complex models such as deep neural networks are even more susceptible to overfitting in these situations (Krizhevsky et al., 2012; Waldeland et al., 2018). This is analogous to underdetermined inverse problems where overfitting the observed data is a common problem. The lack of labeled data makes unsupervised methods more appealing in these situations, but a better approach would be able to directly leverage the few labeled data that are available, without overfitting.

One solution to this overfitting problem is to use *semisupervised learning* (SSL) techniques because they utilize both the labeled *and* unlabeled data in the training process, which is predicted to achieve more accurate labels for the unlabeled data than SL techniques in the context of limited training data (Chapelle et al., 2006; Zhu and Goldberg, 2009). Returning to the inversion analogy, when an inverse problem is underdetermined, a common solution is to stabilize the objective function by adding an additional term that involves the model parameters, otherwise known as regularization (Aster et al., 2005). Regularization smooths the objective function and this prevents the predicted data from overfitting the observed data. Similarly, SSL can be thought of as SL where a regularization term including the unlabeled data is added to the objective function (Zhu and Goldberg, 2009).

The only publication we are aware of that uses semisupervised methods for well log classification is the label propagation (LP) method coupled with self-training given by Dunham et al. (2020). In this previous work, we show that our self-training process of incrementally adding unlabeled points with the highest LP confidence to the labeled dataset can be effective, but a disadvantage of LP is that it is a transductive method. Transductive algorithms operate by learning an internal mapping of the existing labeled and unlabeled instances in order to classify the unlabeled data. As a result, transductive algorithms must be retrained using any additional unlabeled data that labels are sought for (i.e. it is not a classifier). The method we utilize in this paper is semisupervised Gaussian mixture models (ssGMM). This method has similar benefits to our self-training LP technique from Dunham et al. (2020) of being well established and easy to implement, but the unique benefit of ssGMM is that it is an inductive algorithm. Inductive algorithms are designed to make predictions for new unlabeled data without needing to retrain the algorithm. In larger problems, the amount of unlabeled data could number in the millions, and including all of the unlabeled data in training, which would be required of transductive algorithms, may be computationally demanding. Inductive algorithms are advantageous in these situations because they can instead use a subset of the unlabeled data during training and then the learned model can be used to classify the remaining unlabeled data.

We also try to improve our machine learning algorithm predictions through the use of a new hyper-parameter selection strategy. The conventional hyper-parameter selection approach utilizes a cross-validation scheme on the training data and the hyper-parameter combination with the largest mean cross-validation score is used to train the machine learning model (Bishop et al., 1998; Hastie et al., 2009; Krstajic et al., 2014). However, the cross-validation process also produces standard deviations that, to the best of our knowledge, have yet to be directly utilized in standard hyper-parameter selection schemes. The strategy that we introduce here selects a hyper-parameter combination based on simultaneously using the mean and standard deviation scores coming from cross-validation, rather than the default procedure that only relies on the mean cross-validation scores.

We apply these ideas to a lithofacies classification problem where the well log data we use are publicly available through an open competition held in 2016. To assess the performance of the ssGMM method, we compare it to two supervised methods, and our self-training LP method from Dunham et al. (2020). Furthermore, to evaluate the efficacy of our new hyper-parameter selection strategy, we show how algorithm performance (both SL and SSL methods) can vary using our selection strategy versus the conventional approach. This paper is the first to our knowledge that uses a semisupervised Gaussian mixture models method in a well log classification context. We also hope our new hyper-parameter selection strategy will convince our readers to re-evaluate how hyper-parameters are conventionally chosen during training, as well as recognize the applicability of our strategy to other classification problems.

2. Methodology

Machine learning algorithms that possess the capability of assigning classes to unlabeled data fall into two categories: supervised and semisupervised. The objective of supervised learning (SL) is to learn a mapping, otherwise known as a classifier f, from instances (x_i) and their associated classes (y_i) using the training data,

$$L = \{(x_1, y_1), \ldots, (x_l, y_l)\} \tag{1}$$

where l is the number of labeled data. Any given SL algorithm uses the training data (L) to learn a classifier (f), which is modified according to the differences between the predicted and true labels of the training data. Ultimately, the mapping learned from this process is used to make predictions for data where the labels are unknown, i.e. the unlabeled data

$$U = \{x_{l+1}, \ldots, x_{l+u}\} \tag{2}$$

where the actual predictions are given by,

$$H = \{y_{l+1}, \ldots, y_{l+u}\} \tag{3}$$

and u represents the number of unlabeled data. This mapping (f) from supervised learning is analogous to the inverse forward operator ($\mathbf{G}^{-1}$) from inversion, $m = \mathbf{G}^{-1}d$. However, with inversion, this mapping is known and the model is unknown, whereas the model (i.e. the labels L) is known with supervised learning and the mapping is unknown.

SSL algorithms are similar to SL algorithms, but the unlabeled data (U) are incorporated into the training process. As a result, SSL algorithms train with

$$D = \{(x_1, y_1), \ldots, (x_l, y_l), x_{l+1}, \ldots, x_{l+u}\} = L \cup U \tag{4}$$

where D is the union of L and U and the objective is to still make predictions for the unlabeled data (H). From a theoretical perspective, including the unlabeled data in the training process can help SSL algorithms achieve more improved/generalized predictions for U than SL algorithms in the context of low training data.

2.1. Semisupervised Gaussian mixture models (ssGMM)

Many semisupervised techniques are simply extensions of existing unsupervised or supervised methods to include additional information. For instance, semisupervised Gaussian mixture models (ssGMM) use an algorithm that essentially combines a Naïve Bayes classifier (supervised) and Gaussian mixture models (unsupervised). A few different implementations of ssGMM do exist (Nigam et al., 2000; Zhu and Goldberg, 2009; Xing et al., 2013), but we have chosen to follow the approach outlined in Yan et al. (2017). However, none of these implementations provide open-source code for their methods (including Yan et al., 2017), but we have elected to do so (see Computer Code Availability below). The algorithm is summarized below.

In this semisupervised framework, the training data consist of both labeled and unlabeled data (D) and the goal for ssGMM is to employ a probabilistic approach that seeks the labels that maximize the conditional probability $p(D|\theta)$. Training amounts to finding good model parameters (θ), and the maximum likelihood estimate (MLE),

$$\hat{\theta} = \arg\max_{\theta}[p(D|\theta)] = \arg\max_{\theta}[\log p(D|\theta)] \tag{5}$$

gives the parameters under which the data likelihood is the largest. The log-likelihood yields the same maxima as the straight likelihood because log() is monotonic, and using a log-likelihood simplifies the next step considerably. Simplifying log-products into sum-logs and substituting Bayes rule into Eq. (5) gives,

$$\begin{aligned}\log p(D|\theta) &= \log\left(\prod_{i=1}^{l} p(x_i, y_i|\theta)^{\beta} \prod_{i=l+1}^{l+u} p(x_i|\theta)^{1-\beta}\right) \\ &= \beta\sum_{i=1}^{l}\log\left[p(y_i|\theta)p(x_i|y_i,\theta)\right] + (1-\beta)\sum_{i=l+1}^{l+u}\log p(x_i|\theta)\end{aligned} \tag{6}$$

where the first term is the supervised likelihood and the second term is the marginal (unsupervised) likelihood. Eq. (6) is the objective function in general, but if the model parameters are those that describe a multivariate Gaussian distribution for each class, then Eq. (6) becomes,

$$\begin{aligned}\log p(D|\theta) = \beta\sum_{i=1}^{l}\log\left[\pi_{y_i}\mathcal{N}(x_i;\mu_{y_i},\Sigma_{y_i})\right] \\ + (1-\beta)\sum_{i=l+1}^{l+u}\log\left(\sum_{k=1}^{K}\mathcal{N}(x_i;\mu_k,\Sigma_k)\right)\end{aligned} \tag{7}$$

where π, μ, and Σ represent the Gaussian priors, means, and covariances respectively, and $\mathcal{N}$ represents a multivariate normal distribution. The standard implementations of ssGMM (Nigam et al., 2000; Zhu and Goldberg, 2009, Chapter 3) do not include β, but this hyperparameter is introduced by Yan et al. (2017) to give a relative weighting ($0 < \beta < 1$) between the labeled and unlabeled portions of the log-likelihood.

To find the MLE, Eq. (7) needs to be optimized. Unfortunately, the unobserved labels (H) make the log-likelihood non-concave and hard to optimize. The standard remedy is to use the Expectation–Maximization (EM) algorithm (Dempster et al., 1977), which provides an iterative solution to obtaining the MLE in the context of hidden data. For this implementation of ssGMM, the EM algorithm is as follows:

- Step 1: Set $t = 0$ and initialize the model parameters for each class (there are K classes) for a multivariate Gaussian distribution. We achieve this by training a Gaussian Naïve Bayes classifier on the labeled data and obtaining,

$$\theta^{(0)} = \{\pi_k^{(0)}, \mu_k^{(0)}, \Sigma_k^{(0)}\}_{\forall k}. \tag{8}$$

- Step 2: The E-step. Create a matrix γ that is size $n \times K$ where $n = l+u$. For labeled instances ($i = 1, \ldots, l$), define $\gamma_{ik} = 1$ if $y_i = k$, and 0 otherwise. For the unlabeled instances ($i = l+1, \ldots, l+u$), calculate γ_{ik} via,

$$\gamma_{ik} = \frac{\pi_k^{(t)}\mathcal{N}(x_i;\mu_k^{(t)},\Sigma_k^{(t)})}{\sum_j \pi_j^{(t)}\mathcal{N}(x_i;\mu_j^{(t)},\Sigma_j^{(t)})}. \tag{9}$$

- Step 3: The M-step. Determine $\theta^{(t+1)}$ using the current γ_{ik}. For $k = 1, \ldots, K$ compute:

$$NL_k = \sum_{i=1}^{l}\gamma_{ik} \qquad NU_k = \sum_{i=l+1}^{l+u}\gamma_{ik} \qquad C = \beta NL_k + (1-\beta)NU_k$$

$$\pi_k^{(t+1)} = \frac{C}{\beta l + (1-\beta)u}$$

$$\mu_k^{(t+1)} = \frac{\beta\sum_{i=1}^{l}\gamma_{ik}x_i}{C} + \frac{(1-\beta)\sum_{i=l+1}^{l+u}\gamma_{ik}x_i}{C}$$

$$\begin{aligned}\Sigma_k^{(t+1)} = &\frac{\beta\sum_{i=1}^{l}\gamma_{ik}x_i(x_i-\mu_k^{(t+1)})(x_i-\mu_k^{(t+1)})^T}{C} \\ &+\frac{(1-\beta)\sum_{i=l+1}^{l+u}\gamma_{ik}x_i(x_i-\mu_k^{(t+1)})(x_i-\mu_k^{(t+1)})^T}{C}\end{aligned}$$

- Step 4: Increment the iteration step, $t = t + 1$
- Step 5: Repeat Steps 2–4 until Eq. (7) converges to a tolerance defined as the percent change in the log-likelihood (e.g., $tolerance = 0.1$ is one-tenth of a percent)

The matrix γ represents the *soft labels* for the unlabeled data. Once the algorithm converges, a threshold can be applied to assign a hard classification to the unlabeled data points via,

$$y_i = \arg\max_k \gamma_{ik} \qquad \text{for} \quad i = l+1, \ldots, l+u. \tag{10}$$

Implementing this algorithm requires setting the hyper-parameters a priori, which are the tolerance and β. Yan et al. (2017) considers β as the only hyper-parameter, but we find ssGMM is quite sensitive to the tolerance and choose to optimize it rather than set it to a fixed value.

In ssGMM, there is a critical assumption that the labeled data have been generated by multivariate Gaussian distributions and the unlabeled data use the same parametric model for classification. This cluster assumption states that if two points are in the same cluster, then they likely belong to the same class. However, it does not imply that each class forms an isolated cluster, but rather we should not observe data of two distinct classes in the same cluster (Chapelle et al., 2006, Chapter 1). For our implementation, this assumption also implies that each class can only be described by one Gaussian cluster (i.e. it cannot be multinomial). If this cluster assumption is violated in any way (e.g., multinomial, non-Gaussian distributions), then semisupervised learning could actually degrade performance (see Figures 3.2 and 3.3 in Zhu and Goldberg, 2009). Therefore, it is important to ensure that the data features for each class can be described by Gaussian distributions a priori.

2.2. Supervised methods

To assess if the ssGMM method can outperform supervised methods in the context of small training sets on our well log classification scenario, we need supervised methods to serve as a basis of comparison. The first supervised method we consider is a Gaussian Naïve Bayes (GNB) classifier because it represents the fully-supervised version of ssGMM and the output of GNB is the starting condition for ssGMM (see Eq. (8)). Comparing the performance of ssGMM to GNB will also indicate if including the unlabeled data into the training process is beneficial. We use the *GaussianNB* class in `scikit-learn` and implementing this algorithm is trivial because GNB contains no hyper-parameters and no cross-validation is required. For our second supervised method, we use a particular gradient boosting algorithm, *XGBoost* (Chen and Guestrin, 2016), which is the algorithm of choice for the winners of the 2016 Society of Exploration Geophysicists (SEG) machine learning competition (Hall and Hall, 2017). Since we are using the exact same dataset for our study, *XGBoost* is the best supervised method to compare against ssGMM in terms of performance. Unlike GNB, *XGBoost* has several hyper-parameters that require setting, but *XGBoost* has been applied in many different disciplines and is proven

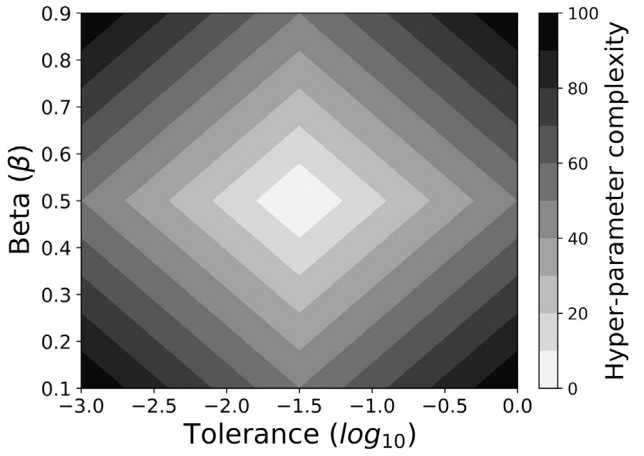

Fig. 1. The assigned hyper-parameter complexity values for the ssGMM method. If a given hyper-parameter selection technique has a conflict, the hyper-parameter combination with the lowest complexity value is chosen. This choice of complexity for ssGMM is made to penalize extreme values for β and the *tolerance*, and favor those that are closer to what we deem to be default values: $(\beta, tolerance) = (0.5, log_{10}[-1.5])$.

to be robust (Tamayo et al., 2016; Torlay et al., 2017; Zhang et al., 2018). We will not discuss the details of these algorithms here, but we refer the interested reader to the literature for GNB (Theodoridis, 2015) and gradient boosting classifiers (Friedman, 2001; Chen and Guestrin, 2016).

2.3. Hyper-parameter selection strategies

Supervised and semisupervised algorithms commonly have hyper-parameters that need to be tuned, and the process of choosing a set of hyper-parameters is called hyper-parameter selection. The selection process first consists of splitting the training data (L) into training and validation sets and then each hyper-parameter setting is evaluated on the validation data using classification metrics such as accuracy, precision, recall, F1, or the area under the curve (AUC) for ROC (receiver operating characteristic) curves (Lever et al., 2016). If only one validation set is used, it is well-known that there is a larger risk of overfitting the training data, which leads to poor prediction accuracies. One remedy is to break the training data into k pieces, or folds, and train the machine learning algorithm with a given hyper-parameter setting on $k-1$ folds, evaluate the performance on the kth fold, and then repeat this process for each fold; this is k-fold cross-validation (CV). For 5-fold CV, for instance, each hyper-parameter setting has a classification score on each of the 5 validation sets. If the data are shuffled and this process is repeated 5 times (i.e. 5-repeated 5-fold CV), then there are 25 classification scores for each hyper-parameter (Krstajic et al., 2014). Conventionally, all of the classification scores are averaged and the standard hyper-parameter selection tactic is to select the hyper-parameter combination with the largest mean CV score (Bishop, 2006, Chapter 1.3; Hastie et al., 2009, Chapter 7.10; Krstajic et al., 2014), and the same can be said for averaging AUC values in multi-class situations (Hand and Till, 2001; Fawcett, 2006). This hyper-parameter combination is then used for training a machine learning model. The hyper-parameter selection functions in `scikit-learn` in Python use the same selection approach, and when there is a conflict (i.e. multiple hyper-parameter combinations with the same mean CV score), the least-complicated combination is selected.

The standard hyper-parameter selection strategy only uses the mean CV scores, but there are also accompanying *standard deviation* CV scores that, to the best of our knowledge, have yet to have been directly leveraged in any selection process. What we propose here is a new selection strategy that *simultaneously* uses the mean and standard deviation CV scores to select a hyper-parameter combination, i.e. a simultaneous mean and standard deviation (SMSD) score,

$$SMSD_j = \alpha \left(\frac{\bar{\boldsymbol{x}}_j - \bar{x}_{CV\,min}}{\bar{x}_{CV\,max} - \bar{x}_{CV\,min}} \right) + (1-\alpha) \left(\frac{\sigma_{CV\,max} - \boldsymbol{\sigma}_j}{\sigma_{CV\,max} - \sigma_{CV\,min}} \right) \quad (11)$$

where $\bar{\boldsymbol{x}}_j$ and $\boldsymbol{\sigma}_j$ are the mean and standard CV scores for the jth hyper-parameter combination, $\bar{x}_{CV\,min}$ and $\bar{x}_{CV\,max}$ are the minimum and maximum mean CV scores on the hyper-parameter grid, $\sigma_{CV\,min}$ and $\sigma_{CV\,max}$ are the minimum and maximum standard deviation CV scores on the hyper-parameter grid, and α gives a relative weight between the mean and standard deviation CV scores. Generally, α can vary on the interval $[0, 1]$, but allowing it to change adds a level of complexity. For the context of this paper, we fix $\alpha = 0.5$ to let the mean and standard deviation scores *equally* contribute to the decision. The hyper-parameter selection scheme using SMSD is simply choosing the hyper-parameter combination with the highest SMSD score, and if there is a conflict, then the least-complicated combination is chosen. For *XGBoost* (XGB), the least-complicated hyper-parameters are those with smaller values, and for ssGMM, this is the hyper-parameter combination with the smallest complexity value as defined in Fig. 1. This scheme is analogous to regularized inverse problems where we not only care about data misfit, but we also care about some measure of the model norm, and we weight the contribution of both these factors in the objective function.

What we are trying to address with the SMSD method is hyper-parameter combinations with the highest mean CV score may not always be optimal. For instance, if a given hyper-parameter combination has the highest mean CV score, but also has the highest standard deviation CV score, is this the optimal choice? Arguably, the optimal choice is one that is both accurate (i.e. high mean) and precise (i.e. low standard deviation) and our SMSD method will select a hyper-parameter combination that has a balance between the mean and standard deviation scores. It is worth mentioning that if the highest mean CV score and the lowest standard deviation CV score align, then the SMSD hyper-parameter choice is equivalent to the default method's choice. Where the hyper-parameter choices from the SMSD and the traditional methods will differ is when the highest mean and the lowest standard deviation CV scores do not align, and what we try to investigate in this paper is if the hyper-parameter combination chosen using the SMSD method trains models that perform better on testing data in these situations.

3. Well log dataset

The well log dataset used for this study is the same dataset that was used for an SEG machine learning competition held in 2016 (Hall, 2016; Hall and Hall, 2017), but the data were ultimately made public by the Kansas Geological Survey. The data consist of ten wells (nine are real, and one is synthetic) drilled in the Hugoton and Panoma fields of southwest Kansas and northwest Oklahoma, and we refer the interested reader to Dubois et al. (2006) for a discussion of the geology of this region. All ten wells contain wire-line log data (i.e. the instances x_i) and core samples (i.e. the associated labels y_i) recorded at half-foot increments for 4137 total data points. Dubois et al. (2006) determine that there are nine lithofacies, or classes, in this dataset and the first three (1–3) are non-marine and the remaining six classes (4–9) are marine lithofacies (see Table 1). The features, or dimensions, for each instance consist of five wire-line logs and two geologic variables interpreted by users. Fig. 2 shows a cross-plot of each of the seven features plotted against each other for the entire dataset. Notice that the NM_M indicator is effective at distinguishing the non-marine from the marine classes, but the classes *within* the non-marine and marine categories are not linearly separable with considerable overlap. It is well-understood that rock units are not always discrete and their physical properties are not unique, and this can lead to poor class separability (Avseth et al., 2005). Misclassifications of this dataset are expected to occur

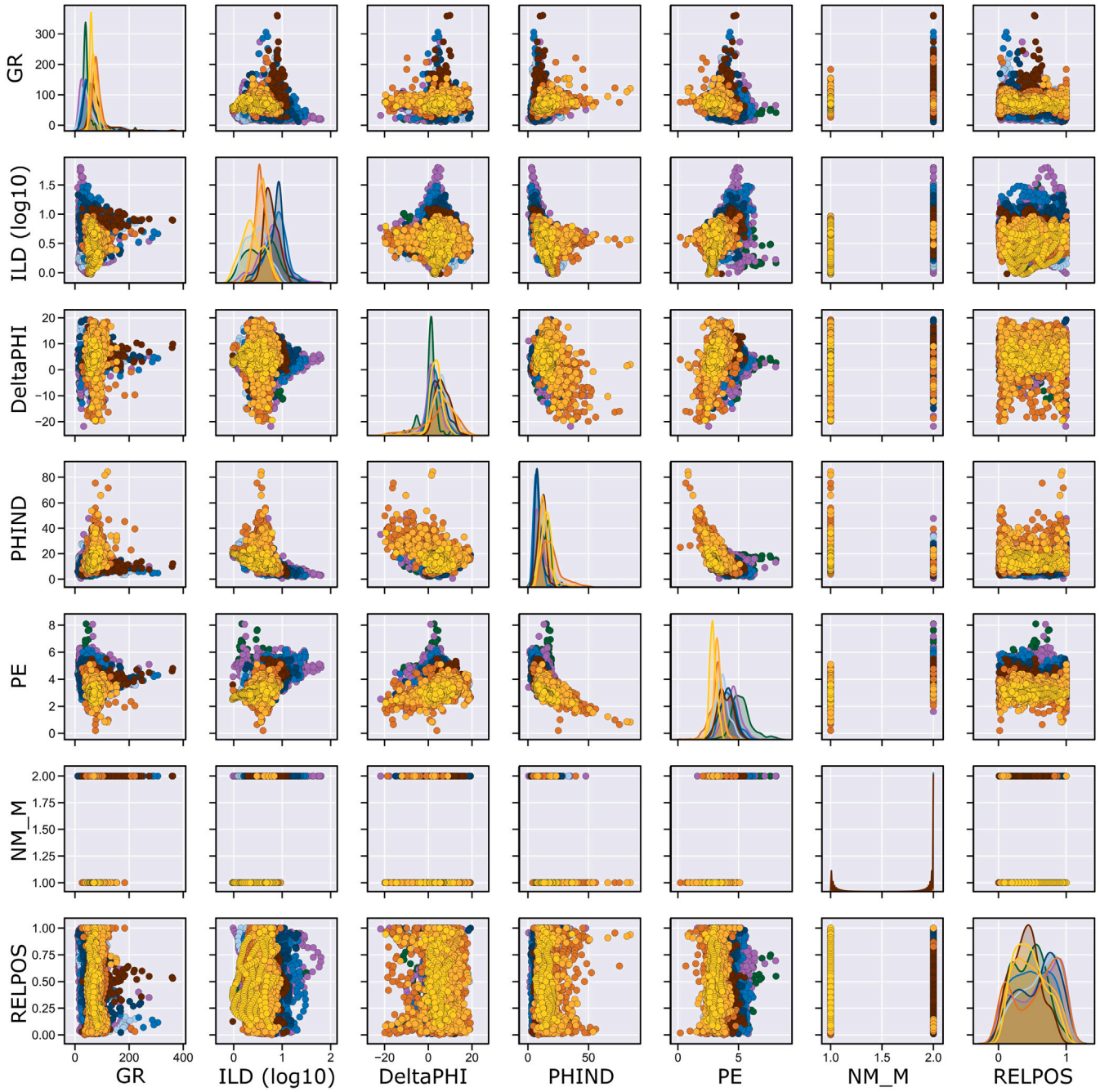

Fig. 2. A cross-plot matrix showing the behavior for all seven data variables with respect to each other for the entire well log dataset. The variable abbreviations represent the following: GR = gamma ray, ILD (log_{10}) = resistivity, DeltaPHI = neutron-density porosity difference, PHIND = average neutron-density porosity, PE = photoelectric effect, NM_M = non-marine/marine indicator, and RELPOS = relative position. The first five features are log variables and the last two features are user interpreted geologic variables. For the classes pertaining to each color, see Table 1. The distributions of each class for each variable are given by the diagonal elements in the matrix, and these distributions indicate the classes have a high degree of overlap.

as a result of this, but one could argue that if a predicted lithofacies is *close* to the true facies, then this could still be classified as correct. For the machine learning competition associated with this dataset, Hall (2016) introduces an *adjacent accuracy* metric that deems lithofacies that occur close to each other depositionally (i.e. via Walthers Law) are also considered correct. These adjacent facies are indicated in Table 1 and we also include this metric.

The machine learning competition using this dataset was structured for the competitors to train a classifier using all ten wells (L) and the competition organizers would test the competitors' classifiers on two additional withheld wells (U), but these extra two wells are not available to the public. As in Dunham et al. (2020), to properly simulate a semisupervised situation with this dataset, we restructure the classification problem so that only one well is used as the labeled data (L) and the objective is to predict the lithofacies for the remaining nine wells (U). However, this one well for training has to be chosen carefully because not all the wells contain every class (i.e. if a class is not present in the training data, then predictions cannot be made for it). For continuity purposes, we choose the same labeled well as Dunham et al. (2020), KIMZEY, and we refer the interested reader to this previous study for the justification of this choice. Fig. 3 depicts the distribution of points for both the labeled and unlabeled data in this situation. The

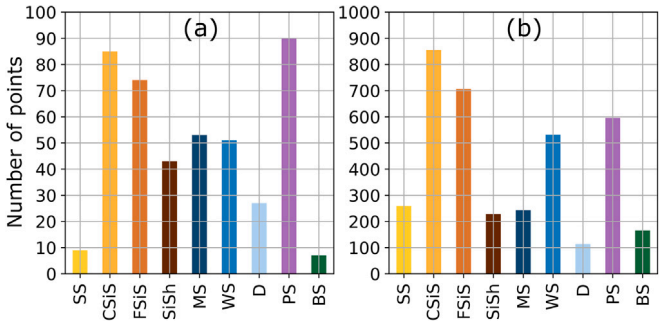

Fig. 3. The distribution of points per class for the (a) one labeled well and the (b) remaining nine unlabeled wells. Classes 1 and 9 have the least number of training points with nine and seven points, respectively.

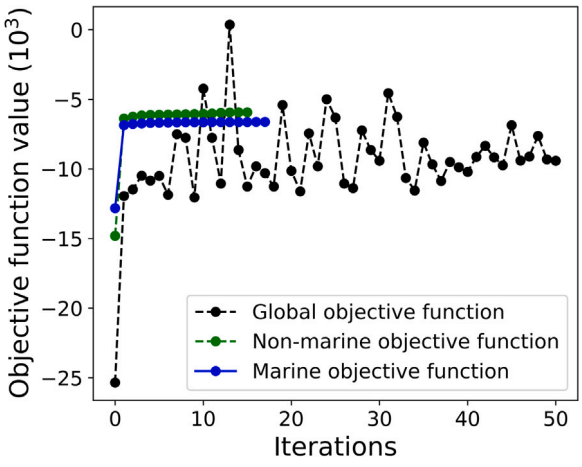

Fig. 4. The behavior of the ssGMM objective function Eq. (7) with respect to the number of iterations when trained on the single labeled well using default hyper-parameter settings ($\beta = 0.50, tolerance = log_{10}[-1.5]$). The ssGMM method does not converge (dashed black line) because the NM_M indicator variable is not Gaussian distributed, but splitting the data into two pieces (non-marine and marine) allows the algorithm to converge.

Table 1
The nine lithofacies for the well log dataset and their corresponding descriptions. The Label column serves as a key for the colors associated with each facies in Figs. 2, 3, 9, and 10. The classes listed in the Adjacent facies column are used to compute the adjacent accuracy metric given by Hall (2016).

Facies	Description	Label	Adjacent facies
Non-marine classes			
1	Sandstone	SS	2
2	Coarse siltstone	CSiS	1,3
3	Fine siltstone	FSiS	2
Marine classes			
4	Siltstone and shale	SiSh	5
5	Mudstone	MS	4,6
6	Wackestone	WS	5,7,8
7	Dolomite	D	6,8
8	Packstone-grainstone	PS	6,7,9
9	Bafflestone	BS	7,8

distributions between the labeled and unlabeled data are similar with minor differences, but what is noteworthy in Fig. 3(a) is that some classes have very few points; the consequences of this are discussed in Section 5.2. Prior to any classification, these data are scaled and normalized using the *RobustScaler* class from `scikit-learn`.

4. Results

4.1. Initial ssGMM test

We begin with a testing stage to investigate how ssGMM behaves when applied to this dataset. No cross-validation procedures are used and we simply train the algorithm on the one labeled well (KIMZEY) using default hyper-parameter values ($\beta = 0.50, tolerance = log_{10}[-1.5]$). This initial test shows that the objective function for ssGMM does not converge (dashed black line in Fig. 4). Recall from Section 2.1 that the inherent cluster assumption of ssGMM is subject to violation if the data cannot be represented by multivariate Gaussians. Notice in Fig. 2 that most of the variables exhibit Gaussian-like distributions, but the exception is the NM_M indicator which is a binary, bimodal variable. Algorithmically, this makes the covariance matrices for each class poorly defined and taking the inverse of these matrices (required to compute $\mathcal{N}$ in Eq. (9)) causes an instability.

Our remedy for this problem, as in Dunham et al. (2020), is to remove the NM_M indicator variable by decomposing the well log data into two separate datasets based on the NM_M indicator. This decomposes the original dataset into non-marine (NM_M = 1) and marine (NM_M = 2) facies datasets[1] that correspond to classes 1–3 and 4–9, respectively (see Table 1). The ssGMM algorithm is trained again (using default hyper-parameters) on the non-marine and marine subsets of KIMZEY, and Fig. 4 shows that the objective function easily converges for both. This is evidence that the NM_M variable is the underlying cause for the ssGMM algorithm not converging on the global data. Moving forward, we apply all algorithms to the separate non-marine and marine datasets.

4.2. Comparison to dunham et al.

Based on our necessity to decompose the well log data into non-marine and marine datasets for ssGMM, we can make direct comparisons here to our previous self-training LP results from Dunham et al. (2020). In this previous work, we considered 3-fold CV with total accuracy as the classification metric to train LP, and we use the same scheme here for ssGMM to make a direct comparison. We mirror the choice of the XGB hyper-parameter grid from Bestagini et al. (2017), but we modify it slightly to achieve better performance for this situation. The hyper-parameter grid we consider for ssGMM is indicated in Fig. 1 with 31 logarithmically spaced values for the $tolerance$ and 17 linearly spaced values for β. We first evaluate the total accuracy of each hyper-parameter choice on the testing data for both non-marine and marine datasets; this will indicate how close future ssGMM hyper-parameter selections are to the maximum achievable accuracies (see Fig. 5). A summary of the results is given in Table 2 where the last row is the self-training label propagation result taken from Table 3 in Dunham et al. (2020). The mean and standard deviation 3-fold CV scores for ssGMM are given in Fig. 6, and the different hyper-parameter selections are indicated by circles. Using the SMSD score Eq. (11) as the hyper-parameter selection strategy is not the focus of this section, but Fig. 6(c) and 6(f) indicate that using the SMSD score gives the same hyper-parameter combinations as the default approach shown in Fig. 6(a) and 6(d). In Table 2, we see ssGMM and self-training label propagation are performing better than GNB and XGB. However, the rows indicating the best possible performance for XGB and ssGMM on the unlabeled data suggest that there is room for improvement for both of these methods.

[1] It is noteworthy that the NM_M indicator is a variable that is created from interpretations of the wireline log variables for the purpose of this dataset (Dubois et al., 2006). Therefore, performing this data splitting procedure on *other* datasets is unlikely, but it is advantageous for ensuring the cluster assumption of ssGMM is not violated in the context of this dataset.

Table 2
Testing data performance for the supervised methods (GNB, XGB) and the semisupervised methods (ssGMM, self-training label propagation) where 3-fold CV with total accuracy as the classification metric is used to train all algorithms. The F-1 and adjacent accuracy metrics are not used for hyper-parameter selection, but are merely presented for comparison. The self-train LP row is taken directly from Dunham et al. (2020). All computations are conducted on a desktop machine (3.2 GHz Intel Core i5 processor) with 16GB RAM and all cross-validations are performed in parallel on four cores. The XGB (best) and ssGMM (best) rows give the highest achievable total accuracy on the non-marine and marine testing data (indicated by the black bulls-eyes in Fig. 5). The models representing the colored cells correspond to the same-colored circles in Figs. 5 and 6.

Machine learning algorithm	Non-marine accuracy (%)	Marine accuracy (%)	Total accuracy (%)	F-1 score (%)	Total adjacent accuracy (%)	Total # of CV fits	Elapsed time
GNB	49.21	32.26	40.64	35.91	82.59	0	0.023 s
XGB	54.40	34.67	44.43	42.33	83.88	700 × 3 × 2	269.3 s
XGB (best)	56.92	38.15	47.43	44.27	84.23	N/A	N/A
ssGMM	55.17	38.09	46.54	43.07	88.37	527 × 3 × 2	136.9 s
ssGMM (best)	58.39	39.59	48.89	45.24	89.21	N/A	N/A
Self-train LP	62.33	39.17	50.62	49.50	88.02	1681 × 3 × 2	9.59 s

Table 3
Testing data performance for XGB and ssGMM using 5-fold and 5-repeated 5-fold cross-validation with total accuracy as the classification metric. Hyper-parameter selection for both algorithms in both situations is achieved using the standard and SMSD (Eq. (11)) approaches. GNB is not included because it contains no hyper-parameters and its performance is the same as shown in Table 2. The models representing the colored cells correspond to the same-colored circles in Figs. 5, 7, and 8. The best performing XGB and ssGMM models are denoted by the *.

Machine learning algorithm	Non-marine accuracy (%)	Marine accuracy (%)	Total accuracy (%)	F-1 score (%)	Total adjacent accuracy (%)	Total # of CV fits	Elapsed time
5-fold cross-validation							
ssGMM	54.84	39.11	46.89	43.09	89.21	527 × 5 × 2	151.46 s
ssGMM (SMSD)	55.27	39.38	47.24	43.39	89.40	527 × 5 × 2	152.31 s
XGB	50.30	34.56	42.34	40.06	82.86	700 × 5 × 2	460.10 s
XGB (SMSD)	50.30	36.33	43.24	40.41	82.88	700 × 5 × 2	458.66 s
5-repeated 5-fold cross-validation							
ssGMM	55.99	33.87	44.81	38.35	84.86	527 × 5 × 5 × 2	520.19 s
ssGMM (SMSD)*	58.28	38.20	48.13	44.53	89.29	527 × 5 × 5 × 2	550.41 s
XGB*	55.27	34.78	44.92	42.63	83.64	700 × 5 × 5 × 2	2188.6 s
XGB (SMSD)	53.75	34.72	44.13	42.35	83.69	700 × 5 × 5 × 2	2190.2 s

4.3. Improving XGB and ssGMM performance through hyper-parameter selection

Only 3-fold CV is considered in Dunham et al. (2020), but we consider a higher fold here to see if improvement can be gained for XGB and ssGMM. We choose to have at minimum one data point from each class in each fold, so the highest fold we can test is 7-fold because Class 9 only has seven points (see Fig. 3). However, 5-fold CV is standard and that is what we test here. Table 3 (top) summarizes the results, and Fig. 7 gives the mean, standard deviation, and SMSD 5-fold CV scores for ssGMM. We see a minor performance improvement for ssGMM using 5-fold rather than 3-fold CV (compare to Table 2), and using the SMSD score gives a marginal improvement over the default hyper-parameter selection for 5-fold CV. Using 5-fold CV for XGB deteriorated the performance slightly compared to using 3-fold CV (see Table 2), but using the SMSD score selected hyper-parameters for XGB that performed slightly better in the 5-fold CV case.

Using 5-fold CV produces five classification metric scores for each hyper-parameter combination to compute the mean and standard deviation CV scores from. However, five values are arguably not enough for the computed mean and standard deviation scores to be statistically significant. A solution to this problem is N-repeated k-fold CV (discussed in Section 2.3) and we consider 5-repeated 5-fold CV so there are 25 classification scores for each hyper-parameter combination. Table 3 (bottom) summarizes these results, and Fig. 8 gives the mean, standard deviation, and SMSD 5-repeated 5-fold CV scores for ssGMM. While the default hyper-parameter selection for ssGMM on the non-marine data performs well, the default for the marine data is quite poor and hinders the overall performance. However, the hyper-parameter choices obtained using the SMSD score vastly improve the performance on both datasets. The default hyper-parameter choices for XGB perform well, but the choices obtained using the SMSD score diminish the performance slightly.

5. Discussion

5.1. Overall performance comparison

One objective of our study is to assess if ssGMM can outperform the considered supervised methods (GNB and XGB) in the context of minimal training data. Recall from Section 2.2 that we consider GNB because it represents the fully-supervised version of ssGMM. Tables 2 and 3 show that ssGMM outperforms GNB in every circumstance and this indicates, for this algorithm, that including the unlabeled data into the training process substantially improves its performance. Our results also demonstrate that ssGMM is able to outperform XGB in terms of accuracy (1%–5% for all classification metrics) and computation time (2-4× faster) in this context. However, the comparison to our previous self-training label propagation method in Section 4.2 indicates that even if we are able to recover the best-possible ssGMM hyper-parameters, this would still not outperform the self-training LP result in either performance or computation time (compare the ssGMM (best) and Self-train LP rows in Table 2). The number of data for this study is quite small, so the benefit of ssGMM being an inductive algorithm is not properly demonstrated here. However, in problems where the number of data are much greater, the inductive capabilities of ssGMM may prove to be more computationally feasible than transductive approaches, such as label propagation.

The second objective of our study is to determine if simultaneously using the mean and standard deviation CV scores (i.e. the SMSD score) to select hyper-parameters is preferred in comparison to the default approach of only using the mean CV scores. For the ssGMM method, Tables 2 and 3 indicate that the SMSD score selects hyper-parameters that perform the same or better than the default selections, and the fact that ssGMM only has two hyper-parameters made the visualization of this quite clear (i.e. Fig. 5–8). Using the SMSD score to select hyper-parameters for XGB gives mixed results. It seems that when the standard hyper-parameter selection for XGB performs poorly, then the SMSD score can select hyper-parameters that perform better (Table 3

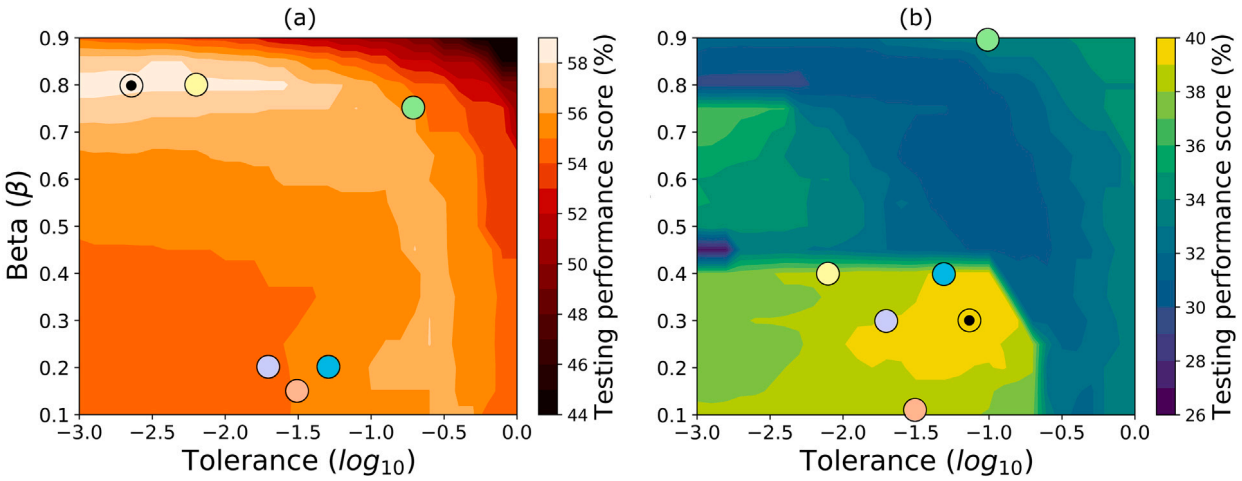

Fig. 5. The total accuracy of each ssGMM hyper-parameter choice on the testing data for the (a) non-marine and (b) marine datasets. This does assume H Eq. (3) is known, which is unrealistic, but these panels help indicate how close the ssGMM models are to the highest accuracy zones. The black bulls-eyes indicate hyper-parameter choices that give the maximum achievable accuracy. The colored circles represent various ssGMM models, and their detailed numerical performances on the nine unlabeled wells are indicated in Tables 2 and 3.

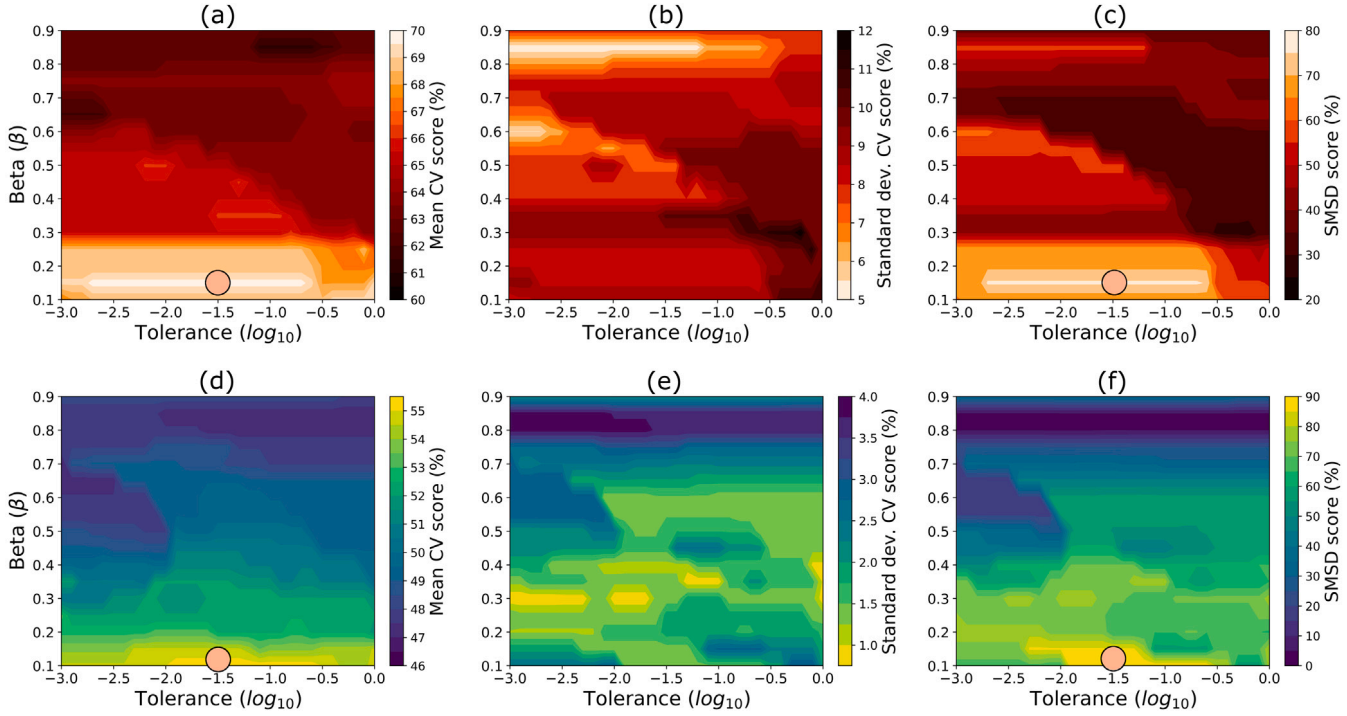

Fig. 6. The mean, standard deviation, and SMSD 3-fold cross-validation scores using ssGMM for the non-marine (a, b, c) and marine (d, e, f) training datasets, respectively. Total accuracy is used as the classification metric. The standard hyper-parameter selection approach selects hyper-parameter combinations indicated by the circles in panels (a) and (d). The SMSD hyper-parameter selection approach selects the hyper-parameter combinations indicated by the circles in panels (c) and (f). For this situation, the chosen hyper-parameter combinations for each approach are identical. See Fig. 5 and Table 2 for the testing performance of the models indicated by circles.

for 5-fold CV). However, if the standard hyper-parameter selection for XGB is already performing well, then using the SMSD score appears to make a poorer selection (Table 3 for 5-repeated 5-fold CV). XGB has many hyper-parameters, and so it is difficult to ascertain the cause of this phenomenon without being able to visualize its performance in the way we do for ssGMM.

5.2. Interpretation

There are a few methods for interpreting these classification results, and the first is inspecting the performance on a per-well basis rather than globally. Table 4 shows the accuracy scores on each of the nine unlabeled wells for the best recovered XGB and ssGMM models. The ssGMM model is able to outperform XGB on seven of the nine unlabeled wells by a notable margin. We can also interpret the classification results by physically observing the facies predictions. The performance of XGB and ssGMM on the SHANKLE and NOLAN wells is characteristic of their overall performance, and so we choose to show the facies predictions for these two wells in Figs. 9 and 10. Notice how in both wells, the XGB prediction is visibly chaotic, which is evidence of overfitting. However, the ssGMM predictions are less chaotic and fit the true facies better, which supports the claim in Section 1 that including unlabeled data in the training process is akin to regularizing underdetermined inverse problems to help prevent overfitting.

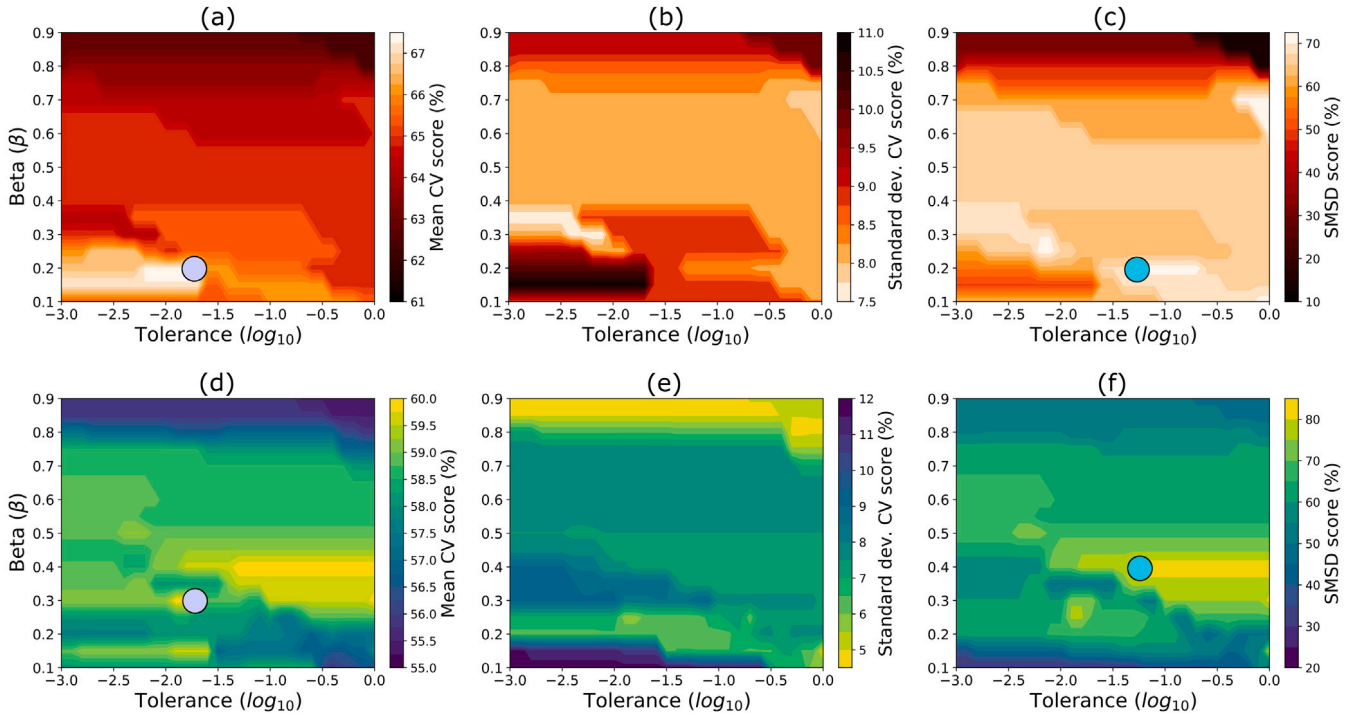

Fig. 7. The mean, standard deviation, and SMSD 5-fold cross-validation scores using ssGMM for the non-marine (a, b, c) and marine (d, e, f) training datasets, respectively. Total accuracy is used as the classification metric. The standard hyper-parameter choices are indicated by the purple circles in panels (a) and (d), and the SMSD hyper-parameter choices are indicated by the cyan circles in panels (c) and (f). For this situation, the SMSD hyper-parameters give a slightly better performance on the testing data (see Fig. 5 and Table 3).

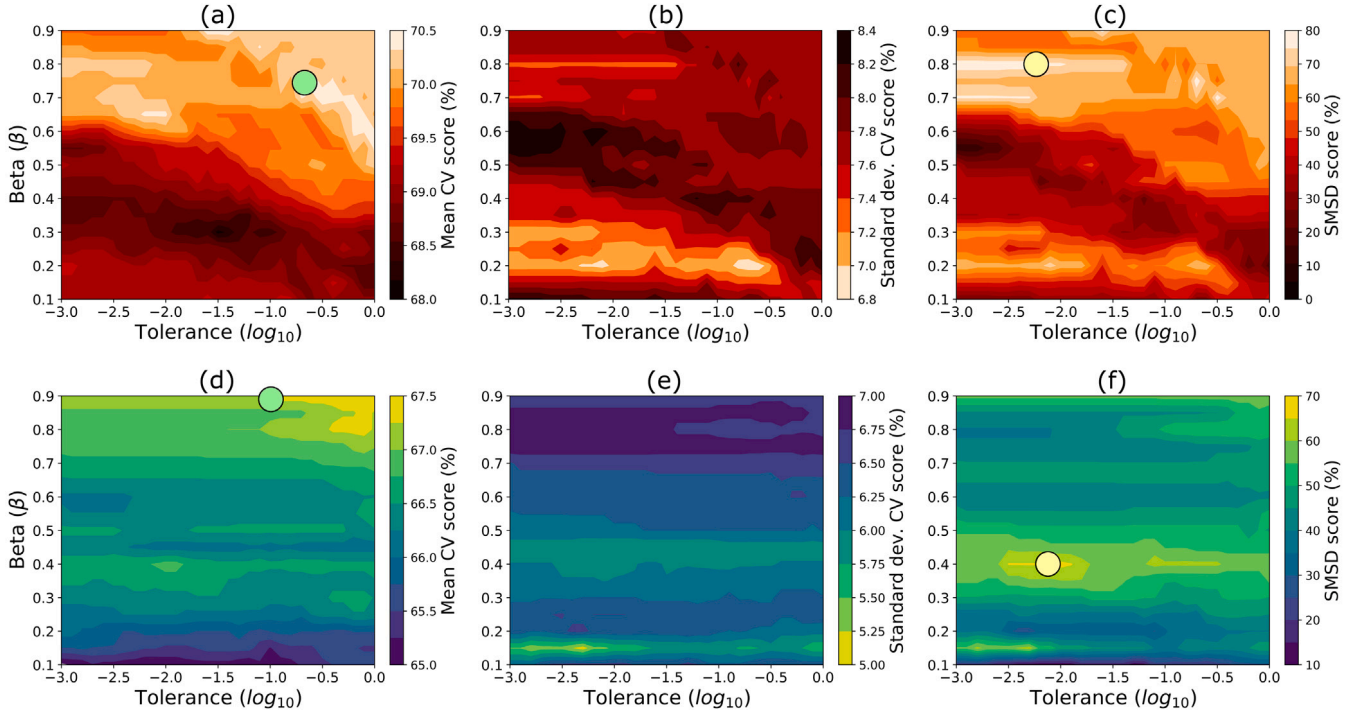

Fig. 8. The mean, standard deviation, and SMSD 5-repeated 5-fold cross-validation scores using ssGMM for the non-marine (a, b, c) and marine (d, e, f) training datasets, respectively. Total accuracy is used as the classification metric. The standard hyper-parameter choices are indicated by the green circles in panels (a) and (d), and the SMSD hyper-parameter choices are indicated by the yellow circles in panels (c) and (f). For this situation, the standard hyper-parameter choice for the non-marine data (a) performs well on the testing data, but the standard hyper-parameter choice for the marine data (d) does not. However, the SMSD score chooses standard hyper-parameters that significantly improve the testing performance of both the non-marine and marine datasets. See Fig. 5 and Table 3 for the testing performance of the models indicated by circles.

The maximum probabilities used to classify the unlabeled data for ssGMM (Eq. (10)) are given as the probability logs in Figs. 9 and 10. These probabilities are a benefit to ssGMM and they can be useful for interpretation. For instance, the probability tends to drop at predicted lithofacies interfaces, which supports the well-understood notion that rock unit boundaries are not always discrete.

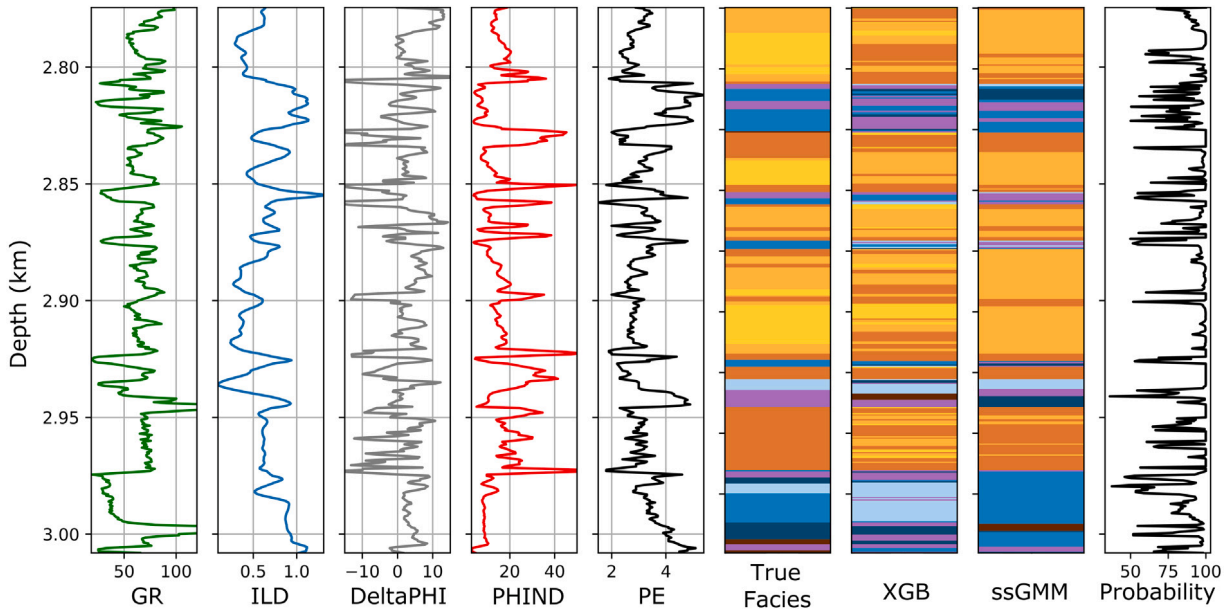

Fig. 9. A comparison of the XGB and ssGMM facies predictions to the true facies for the unlabeled well, SHANKLE. For the performance of these models on SHANKLE, see Table 4. The five log variables are shown for reference. The final column gives the probabilities for the ssGMM predicted facies. See Table 1 for the facies colors key.

Table 4
The prediction accuracies for the best XGB and ssGMM models decomposed into individual accuracies for each of the unlabeled wells. Table 3 denotes the best models for XGB and ssGMM come from 5-repeated 5-fold CV. The total accuracies provided in the last row correspond to those given in Table 3 and are calculated by taking the weighted average of the individual accuracies for each well.

Unlabeled well name	# points	XGB accuracy	ssGMM accuracy
SHANKLE	449	44.77	52.78
CROSS H CATTLE	501	28.94	32.34
NEWBY	463	41.04	44.28
LUKE	461	60.52	64.43
CHURCHMAN BIBLE	404	42.08	40.84
ALEXANDER	466	52.36	53.65
SHIMPLIN	471	51.59	54.99
NOLAN	415	44.34	49.40
RECRUIT F9	68	7.35	0.00
Total	3698	44.92	48.13

Another technique for visualizing and interpreting the classification performance is through confusion matrices. Fig. 11 shows the confusion matrices for the best recovered XGB and ssGMM models. Notice how the predictions cluster closer to the diagonal for ssGMM compared to XGB; this is reflected by the higher accuracy and adjacent accuracy as indicated in Table 3. The classification ability of XGB for Classes 1 and 9 is poor (<10%), but ssGMM is unable to predict for these two classes at all. Classes 1 and 9 only have nine and seven points respectively in the training data (Fig. 3a) and the initial covariance matrices describing each class, which are 6-dimensional, are poorly constrained and ill-conditioned because of this. In Table 4, the RECRUIT F9 well is a synthetic well that only contains Class 9 and this is why ssGMM has an accuracy of 0% on that well. Table 4 also shows that the prediction performance for the CROSS H CATTLE well is quite poor, and this is because CROSS H CATTLE has many data points associated with Class 1. While the training data for other classes are sufficient to define their corresponding covariance matrices, more data are certainly needed to constrain the covariance matrices for Classes 1 and 9.

5.3. Comparison to machine learning competition

Upon inspecting the performance of all the algorithms in Section 4, it is apparent that the prediction accuracies on the unlabeled data are quite low (< 50%). Our explanation for the absolute accuracies being so low is because the classes are poorly separated (see Fig. 2), and we come to this conclusion in our previous work (Dunham et al., 2020). In comparison, however, all the winners of the SEG machine learning competition were able to achieve F1-scores of 62%–64% on the two withheld wells using XGB (Hall and Hall, 2017), whereas our implementation of XGB only roughly achieves an F1-score of 42% on the nine unlabeled wells. This discrepancy can be attributed to three factors: the amount of training data, filtered outputs, and feature expansion. For our semisupervised scenario, we are only training with roughly one-tenth of the data that was used for the competition. The competition winners also apply a median filter to their XGB predictions (Bestagini et al., 2017); we do not do this because we feel an algorithm's prediction is best represented by its raw output. Lastly, all the winners use feature expansion, a process of artificially generating more features to help improve class separability, to improve their results. For the competition problem, feature expansion was quite effective and improved F1-scores by 5%–6% (Bestagini et al., 2017). However, feature expansion does not work for our semisupervised scenario because some classes are poorly constrained (e.g., seven and nine points respectively for Classes 1 and 9) and representing these classes in higher dimensions makes them even less constrained; this is a consequence of the curse of dimensionality. We do not show these results for brevity, but the performance for both ssGMM and XGB diminishes when we use similar feature expansion techniques as those from the competition. Together, these three factors are responsible for the disparity between our results and those from the competition.

6. Conclusion

The two objectives of this paper are to investigate (1) if semisupervised Gaussian mixture models (ssGMM) can outperform a widely-used supervised algorithm, XGBoost (XGB), in the context of a well log classification example with limited training data, and (2) if our new hyper-parameter selection strategy that simultaneously uses the mean and standard deviation (SMSD) cross-validation scores can make better selections than the default approach. The results first show that one of the well log data features violates the Gaussian assumption, which causes ssGMM to not converge. Our remedy for this situation is decomposing the dataset into two pieces to remove this feature. This

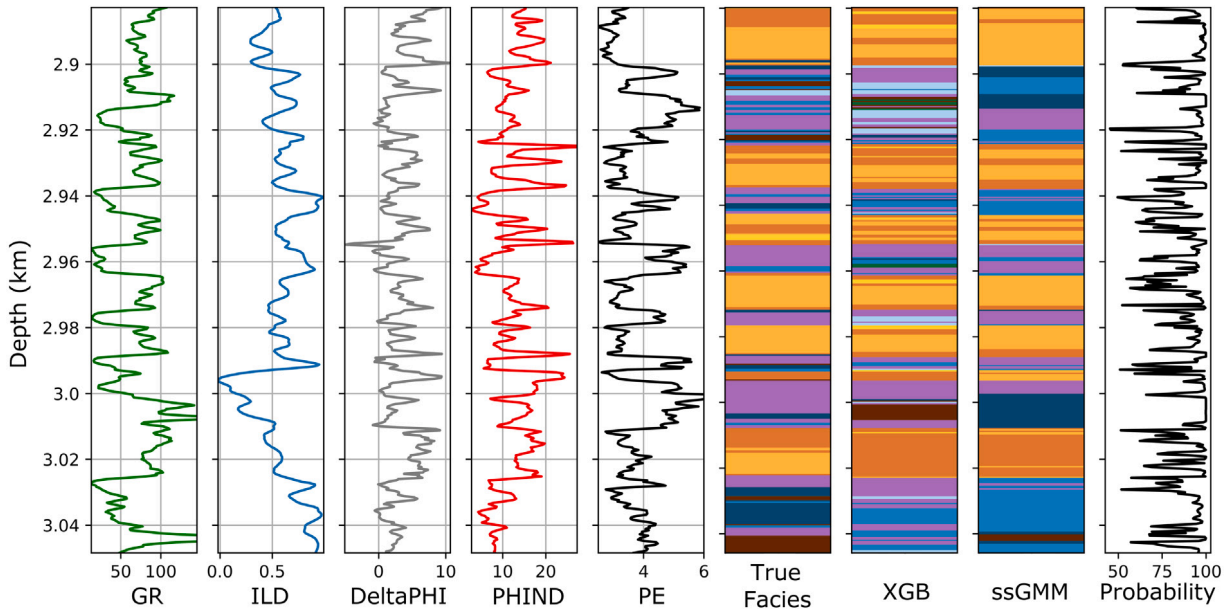

Fig. 10. A comparison of the XGB and ssGMM facies predictions to the true facies for the unlabeled well, NOLAN. For the performance of these models on NOLAN, see Table 4. The five log variables are shown for reference. The final column gives the probabilities for the ssGMM predicted facies. See Table 1 for the facies colors key.

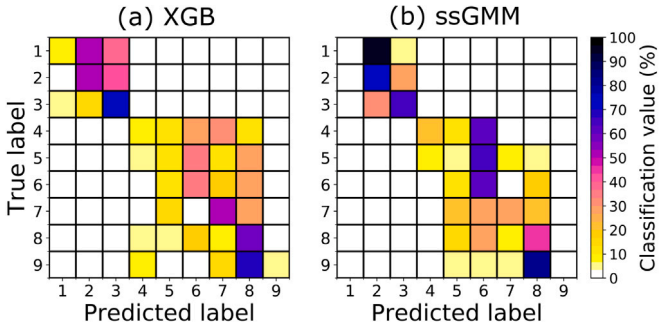

Fig. 11. The normalized confusion matrices for the (a) default XGB 5-repeated 5-fold CV model and the (b) ssGMM 5-repeated 5-fold CV model selected using the SMSD score (i.e. the best models for XGB and ssGMM indicated by the * in Table 3). The predictions shown by these matrices are for the nine unlabeled wells. Diagonal cells represent correct classification, off-diagonal cells represent misclassification.

demonstrates how important it is to perform tests prior to classification to ensure that the data are not violating any underlying assumptions. Once this procedure is performed, our results demonstrate that the ssGMM method is able to outperform XGB, but not by a significant margin. However, a benefit of ssGMM is that it is a simple semisupervised algorithm (i.e. only two hyper-parameters) that can still achieve a better performance than a complex supervised algorithm, such as XGB. Using our new proposed SMSD score for hyper-parameter selection gives promising results for ssGMM, but mixed results for XGB. However, we only compare the performance of the default and SMSD hyper-parameter selection strategies using one dataset and two algorithms, and so future tests using different algorithms and/or datasets would help determine the true efficacy of the SMSD hyper-parameter selection strategy. Nonetheless, our visualization of the lithofacies predictions supports the well-known claim that supervised methods are prone to overfitting when the training data are minimal, but including the unlabeled data into the training process (i.e. semisupervised learning) mitigates this phenomenon.

CRediT authorship contribution statement

Michael W. Dunham: Conceptualization, Methodology, Software, Formal analysis, Validation, Visualization, Writing - original draft. **Alison Malcolm:** Supervision, Writing - review & editing, Funding acquisition. **J. Kim Welford:** Supervision, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Computer code availability

- *Name of code*: `ssGMM`
- *Developer*: Michael W. Dunham
- *Contact*: Department of Earth Sciences, Memorial University of Newfoundland, St. Johns NL A1B 3X5, Canada; mwdunham@mun.ca
- *First release*: 2019
- *Hardware required*: tests were performed on a standard workstation with the following specifications: Intel i5-4750 (4 CPUs) 3.20 GHz processor + 16 GB RAM
- *Software required*: core Python 3 modules: `scikit-learn`, `joblib`, `numpy`, `pandas`, `matplotlib`.
- *Programming language*: the code is written in Python 3
- *How to access the source code*: the source file for the `ssGMM` program, data files, and accompanying Jupyter notebooks that reproduce results from this paper are included in the following public repository: https://github.com/mwdunham/ssGMM

Acknowledgments

The authors would like to thank the anonymous reviewers for their comments, ComputeCanada for their computing support, and also Chevron, the Natural Sciences and Engineering Research Council of Canada, and InnovateNL, Canada for their financial support.

References

Al-Bulushi, N., King, P.R., Blunt, M.J., Kraaijveld, M., 2009. Development of artificial neural network models for predicting water saturation and fluid distribution. J. Petrol. Sci. Eng. 68 (4), 197–208. http://dx.doi.org/10.1016/j.petrol.2009.06.017.

Al-Bulushi, N.I., King, P.R., Blunt, M.J., Kraaijveld, M., 2012. Artificial neural networks workflow and its application in the petroleum industry. Neural Comput. Appl. 21 (3), 409–421. http://dx.doi.org/10.1007/s00521-010-0501-6.

Aster, R.C., Borchers, B., Thurber, C.H., 2005. Parameter Estimation and Inverse Problems. Elsevier Academic Press.

Avseth, P., Mukerji, T., Mavko, G., 2005. Quantitative Seismic Interpretation: Applying Rock Physics Tools to Reduce Interpretation Risk. Cambridge University Press.

Baldwin, J.L., Bateman, R.M., Wheatley, C.L., 1990. Application of a neural network to the problem of mineral identification from well-logs. Log Anal. 31 (5), 279–293.

Benaouda, D., Wadge, G., Whitmarsh, R.B., Rothwell, R.G., MacLeod, C., 1999. Inferring the lithology of borehole rocks by applying neural network classifiers to downhole logs: an example from the Ocean Drilling Program. Geophys. J. Int. 136 (2), 477–491. http://dx.doi.org/10.1046/j.1365-246X.1999.00746.x.

Bestagini, P., Lipari, V., Tubaro, S., 2017. A machine learning approach to facies classification using well logs. In: SEG Technical Program Expanded Abstracts 2017. Society of Exploration Geophysicists, pp. 2137–2142. http://dx.doi.org/10.1190/segam2017-17729805.1.

Bishop, C.M., 2006. Pattern recognition and machine learning. In: Jordan, M., Kleinburg, J., Scholköpf, B. (Eds.), In: Information Science and Statistics, Springer-Verlag.

Bishop, C.M., Svensén, M., Williams, C.K.I., 1998. The generative topographic mapping. Neural Comput. 10 (1), 215–234. http://dx.doi.org/10.1162/089976698300017953.

Chapelle, O., Schölkopf, B., Zien, A., 2006. Semi-Supervised Learning, first ed. MIT Press, Cambridge, MA.

Chen, T., Guestrin, C., 2016. XGBoost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, San Francisco, California, USA, pp. 785–794. http://dx.doi.org/10.1145/2939672.2939785.

Dempster, A.P., Laird, N.M., Rubin, D.B., 1977. Maximum likelihood from incomplete data via the EM algorithm. J. R. Stat. Soc. 39 (1), 1–38.

Dubois, M.K., Bohling, G.C., Chakrabarti, S., 2007. Comparison of four approaches to a rock facies classification problem. Comput. Geosci. 33 (5), 599–617. http://dx.doi.org/10.1016/j.cageo.2006.08.011.

Dubois, M.K., Byrnes, A.P., Bohling, G.C., Doveton, J.H., 2006. Multiscale geologic and petrophysical modeling of the giant Hugoton Gas Field (Permian), Kansas and Oklahoma, U.S.A.. In: Harris, P.M., Weber, L.J. (Eds.), Giant Hydrocarbon Reservoirs of the World: From Rocks To Reservoir Characterization and Modeling. AAPG Memoir 88/SEPM Special Publication, pp. 307–353.

Dunham, M.W., Malcolm, A., Welford, J.K., 2020. Improved well-log classification using semisupervised label propagation and self-training, with comparisons to popular supervised algorithms. Geophysics 85 (1), O1–O15. http://dx.doi.org/10.1190/GEO2019-0238.1.

Fawcett, T., 2006. An introduction to ROC analysis. Pattern Recognit. Lett. 27 (8), 861–874. http://dx.doi.org/10.1016/j.patrec.2005.10.010.

Feng, R., Luthi, S.M., Gisolf, D., Angerer, E., 2018a. Reservoir lithology classification based on seismic inversion results by hidden Markov models: Applying prior geological information. Mar. Pet. Geol. 93, 218–229. http://dx.doi.org/10.1016/j.marpetgeo.2018.03.004.

Feng, R., Luthi, S.M., Gisolf, D., Angerer, E., 2018b. Reservoir lithology determination by hidden Markov random fields based on a Gaussian mixture model. IEEE Trans. Geosci. Remote Sens. 56 (11), 6663–6673. http://dx.doi.org/10.1109/TGRS.2018.2841059.

Friedman, J.H., 2001. Greedy function approximation: A gradient boosting machine. Ann. Statist. 29 (5), 1189–1232, URL: http://www.jstor.org/stable/2699986.

Gallop, J., 2006. Facies probability from mixture distributions with non-stationary impedance errors. In: SEG Technical Program Expanded Abstracts 2006. Society of Exploration Geophysicists, pp. 1801–1805. http://dx.doi.org/10.1190/1.2369874.

Grana, D., Lang, X., Wu, W., 2017. Statistical facies classification from multiple seismic attributes: comparison between Bayesian classification and expectation-maximization method and application in petrophysical inversion. Geophys. Prospect. 65 (2), 544–562. http://dx.doi.org/10.1111/1365-2478.12428.

Hall, B., 2016. Facies classification using machine learning. Lead. Edge 35 (10), 906–909. http://dx.doi.org/10.1190/tle35100906.1.

Hall, M., Hall, B., 2017. Distributed collaborative prediction: Results of the machine learning contest. Lead. Edge 36 (3), 267–269. http://dx.doi.org/10.1190/tle36030267.1.

Hand, D.J., Till, R.J., 2001. A simple generalisation of the area under the ROC curve for multiple class classification problems. Mach. Learn. 45 (2), 171–186. http://dx.doi.org/10.1023/A:1010920819831.

Hardisty, R., Wallet, B.C., 2017. Unsupervised seismic facies from mixture models to highlight channel features. In: SEG Technical Program Expanded Abstracts 2017. Society of Exploration Geophysicists, pp. 2289–2293. http://dx.doi.org/10.1190/segam2017-17794438.1.

Hastie, T., Tibshirani, R., Friedman, J., 2009. The elements of statistical learning, second ed. Springer Series in Statistics, Springer-Verlag, http://dx.doi.org/10.1007/978-0-387-84858-7.

Keynejad, S., Sbar, M.L., Johnson, R.A., 2019. Assessment of machine-learning techniques in predicting lithofluid facies logs in hydrocarbon wells. Interpretation 7 (3), SF1–SF13. http://dx.doi.org/10.1190/INT-2018-0115.1.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (Eds.), Advances in Neural Information Processing Systems, vol. 25. Curran Associates, Inc., pp. 1097–1105.

Krstajic, D., Buturovic, L.J., Leahy, D.E., Thomas, S., 2014. Cross-validation pitfalls when selecting and assessing regression and classification models. J. Cheminform. 6 (10), 1–15. http://dx.doi.org/10.1186/1758-2946-6-10.

Lever, J., Krzywinski, M., Altman, N., 2016. Point of significance: classification evaluation. Nature Methods 13 (8), 603–604. http://dx.doi.org/10.1038/nmeth.3945.

Li, Y., Anderson-Sprecher, R., 2006. Facies identification from well logs: A comparison of discriminant analysis and Naïve Bayes classifier. J. Petrol. Sci. Eng. 53, 149–157. http://dx.doi.org/10.1016/j.petrol.2006.06.001.

Lindberg, D.V., Grana, D., 2015. Petro-elastic log-facies classification using the Expectation-Maximization algorithm and Hidden Markov Models. Math. Geosci. 47, 719–752. http://dx.doi.org/10.1007/s11004-015-9604-z.

Maiti, S., Tiwari, R.K., Kümpel, H.-J., 2007. Neural network modelling and classification of lithofacies using well log data: A case study from KTB borehole site. Geophys. J. Int. 169 (2), 733–746. http://dx.doi.org/10.1111/j.1365-246X.2007.03342.x.

Malvić, T., Velić, J., Horváth, J., Cvetković, M., 2010. Neural networks in petroleum geology as interpretation tools. Cent. Eur. Geol. 53 (1), 97–115. http://dx.doi.org/10.1556/CEuGeol.53.2010.1.6.

McCormack, M.D., 1991. Neural computing in geophysics. Lead. Edge 10 (1), 11–15. http://dx.doi.org/10.1190/1.1436771.

Nigam, K., McCallum, A.K., Thrun, S., Mitchell, T., 2000. Text classification from labeled and unlabeled documents using EM. Mach. Learn. 39 (2), 103–134. http://dx.doi.org/10.1023/A:100769271.

Rogers, S.J., Fang, J.H., Karr, C.L., Stanley, D.A., 1992. Determination of lithology from well logs using a neural network. AAPG Bull. 76 (5), 731–739.

Saggaf, M.M., Nebrija, E.L., 2000. Estimation of lithologies and depositional facies from wire-line logs. AAPG Bull. 84 (10), 1633–1646.

Tamayo, D., Silburt, A., Valencia, D., Menou, K., Ali-Dib, M., Petrovich, C., Huang, C.X., Rein, H., van Laerhoven, C., Paradise, A., Obertas, A., Murray, N., 2016. A machine learns to predict the stability of tightly packed planetary systems. Astrophys. J. Lett. 832 (2), 1–5. http://dx.doi.org/10.3847/2041-8205/832/2/L22.

Theodoridis, S., 2015. Machine Learning: A Bayesian and Optimization Perspective. Academic Press Inc.

Torlay, L., Perrone-Bertolotti, M., Thomas, E., Baciu, M., 2017. Machine learning - XGBoost analysis of language networks to classify patients with epilepsy. Brain Inform. 4 (3), 159–169. http://dx.doi.org/10.1007/s40708-017-0065-7.

Waldeland, A.U., Jensen, A.C., Gelius, L.-J., Solberg, A.H.S., 2018. Convolutional neural networks for automated seismic interpretation. Lead. Edge 37 (7), 529–537. http://dx.doi.org/10.1190/tle37070529.1.

Wallet, B.C., Hardisty, R., 2019. Unsupervised seismic facies using Gaussian mixture models. Interpretation 7 (3), SE93–SE111. http://dx.doi.org/10.1190/INT-2018-0119.1.

Wang, G., Carr, T.R., Ju, Y., Li, C., 2013. Identifying organic-rich Marcellus Shale lithofacies by support vector machine classifier in the Appalachian Basin. Comput. Geosci. 64, 52–60. http://dx.doi.org/10.1016/j.cageo.2013.12.002.

Xing, X., Yu, Y., Jiang, H., Du, S., 2013. A multi-manifold semi-supervised Gaussian mixture model for pattern classification. Pattern Recognit. Lett. 34 (16), 2118–2125. http://dx.doi.org/10.1016/j.patrec.2013.08.005.

Yan, H., Zhou, J., Pang, C.K., 2017. Gaussian mixture model using semisupervised learning for probabilistic fault diagnosis under new data categories. IEEE Trans. Instrum. Meas. 66 (4), 723–733. http://dx.doi.org/10.1109/TIM.2017.2654552.

Zhang, D., Qian, L., Mao, B., Huang, C., Huang, B., Si, Y., 2018. A data-driven design for fault detection of wind turbines using random forests and XGboost. IEEE Access 6, 21020–21031. http://dx.doi.org/10.1109/ACCESS.2018.2818678.

Zhu, X., Goldberg, A.B., 2009. Introduction to semi-supervised learning. In: Brachman, R.J., Dietterich, T. (Eds.), Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool, http://dx.doi.org/10.2200/S00196ED1V01Y200906AIM006.