GEOPHYSICS, VOL. 83, NO. 5 (SEPTEMBER-OCTOBER 2018); P. O83–O95, 7 FIGS., 4 TABLES.
10.1190/GEO2017-0595.1

Seismic facies analysis using machine learning

Thilo Wrona[1], Indranil Pan[2], Robert L. Gawthorpe[1], and Haakon Fossen[1]

ABSTRACT

Seismic interpretations are, by definition, subjective and often require significant time and expertise from the interpreter. We are convinced that machine-learning techniques can help address these problems by performing seismic facies analyses in a rigorous, repeatable way. For this purpose, we use state-of-the-art 3D broadband seismic reflection data of the northern North Sea. Our workflow includes five basic steps. First, we extract seismic attributes to highlight features in the data. Second, we perform a manual seismic facies classification on 10,000 examples. Third, we use some of these examples to train a range of models to predict seismic facies. Fourth, we analyze the performance of these models on the remaining examples. Fifth, we select the "best" model (i.e., highest accuracy) and apply it to a seismic section. As such, we highlight that machine-learning techniques can increase the efficiency of seismic facies analyses.

INTRODUCTION

Seismic reflection data are a key source of information in numerous fields of geoscience, including sedimentology and stratigraphy (e.g., Vail, 1987; Posamentier, 2004), structural geology (Baudon and Cartwright, 2008; Jackson et al., 2014), geomorphology (e.g., Posamentier and Kolla, 2003; Cartwright and Huuse, 2005; Bull et al., 2009), and volcanology (e.g., Hansen et al., 2004; Planke et al., 2005; Magee et al., 2013). However, the often subjective and nonunique interpretation of seismic reflection data has led to long-standing debates based on contrasting geologic interpretations of the same or similar data sets (e.g., Stewart and Allen, 2002; Underhill, 2004). Moreover, seismic interpretations require significant amounts of time, experience, and expertise from interpreters (e.g., Bond et al., 2012; Bond, 2015; Macrae et al., 2016). We believe that machine-learning techniques can help interpreters reduce some of these problems associated with seismic facies analyses.

Machine learning describes a set of computational methods that are able to learn from data to make accurate predictions. Previous applications of machine learning to seismic reflection data focus on the detection of geologic structures, such as faults and salt bodies (e.g., Hale, 2013; Zhang et al., 2014; Guillen et al., 2015; Araya-Polo et al., 2017; Huang et al., 2017) and unsupervised seismic facies classification, in which an algorithm chooses the number and types of facies (e.g., Coléou et al., 2003; de Matos et al., 2006). Although early studies primarily used clustering algorithms to classify seismic data (e.g., Barnes and Laughlin, 2002; Coléou et al., 2003), recent studies focus on the application of artificial neural networks (e.g., de Matos et al., 2006; Huang et al., 2017). To demonstrate the strength of these advanced algorithms, this study compares 20 different classification algorithms (e.g., K-nearest neighbor, support vector machines, and artificial neural networks).

Although unsupervised classification algorithms are, in theory, able to identify the main seismic facies in a given data set, in practice, it can be difficult to correlate these automatically classified facies to existing geologic units. This correlation can be done by visualizing facies in a lower dimensional space (e.g., Gao, 2007) or self-organized maps (e.g., Coléou et al., 2003). As an alternative, we introduce a simple supervised machine-learning workflow, in which the user can define the number and type of seismic facies used for classification. This approach avoids the correlation by allowing the user to adapt the workflow to a given problem, in which seismic facies can be based on existing geologic units.

To demonstrate the advantages of this approach, we describe the application of supervised machine learning to a seismic facies analysis using 3D broadband seismic reflection data of the northern North

Manuscript received by the Editor 5 September 2017; revised manuscript received 27 February 2018; published ahead of production 11 June 2018; published online 05 September 2018.
[1]University of Bergen, Department of Earth Science, Allégaten 41, N-5007 Bergen, Norway. E-mail: thilo.wrona@uib.no; rob.gawthorpe@uib.no; haakon.fossen@uib.no.
[2]Imperial College, Department of Earth Science and Engineering, Prince Consort Road, London SW7 2BP, UK. E-mail: indranilpan@gmail.com.

DOI:10.1190/geo2017-0595.1

Sea. Our workflow consists of five basic steps. First, we extract features from the data by calculating 15 seismic attributes. Second, we generate training data by manually sorting 10,000 examples into four facies. Third, we train 20 models to classify seismic facies using some of these examples. Fourth, we assess the performance of these models using the remaining examples. Fifth, we select the best model based on its performance and apply it to a seismic section. Our results demonstrate that machine-learning algorithms are able to perform seismic facies analyses, which are crucial for mapping sedimentary sequences, structural elements, and fluid contacts.

3D SEISMIC REFLECTION DATA

This study uses state-of-the-art 3D broadband seismic reflection data (CGG BroadseisTM) of the northern North Sea (Figure 1). The data covers an area of $35,410\ \text{km}^2$ and was acquired using a series of up to 8-km-long streamers towed ~40 m deep. The data recording extends to 9 s with a time sampling of 4 ms. Broadseis data covers a wide range of frequencies reaching from 2.5 to 155 Hz (Firth et al., 2014). The binning size was 12.5×18.75 m. The data was 3-D true amplitude Kirchhoff prestack time migrated. The seismic volume was zero-phase processed with SEG normal polarity; i.e., a positive reflection (white) corresponds to an acoustic-impedance increase with depth.

MANUAL INTERPRETATION

Supervised machine learning requires a subset of the data for training and testing models. We therefore select a reasonable number (10,000) of examples to perform a manual seismic facies classification. This number represents a trade-off between the time required for manual classification and the achieved model accuracy (>0.95). After testing different sizes (from 10×10 to 500×500 samples), we selected an example size of 100×100 samples (Figure 2), which results in high model accuracies (>0.95). The classification follows standard

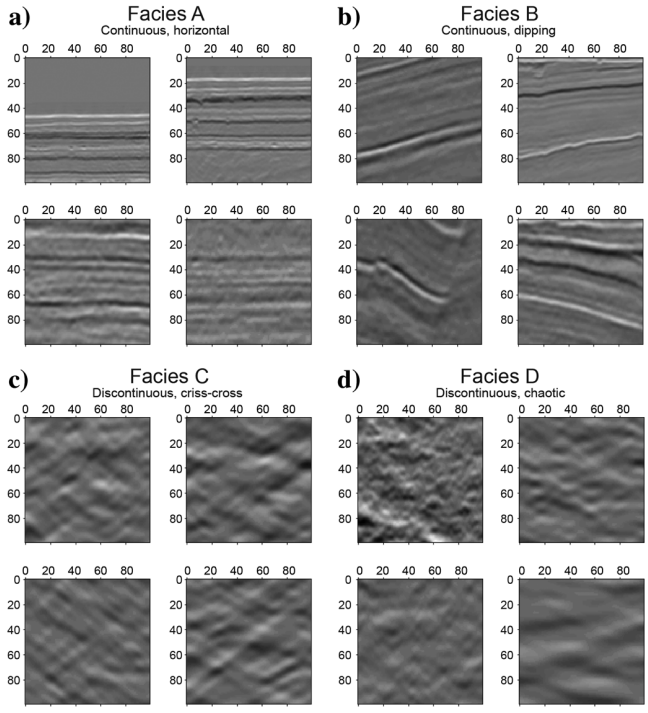

Figure 2. Representative examples of manual seismic facies classification of the four seismic facies chosen for this study: (a) continuous horizontal reflectors, (b) continuous dipping reflectors, (c) discontinuous crisscrossing reflectors, and (d) discontinuous chaotic reflectors. The horizontal axes show distance in m, and the vertical axes show two-way traveltime in ms. The number of examples (10,000) is balanced across classes with each of the four classes containing the same number of examples (2500). Seismic data courtesy of CGG.

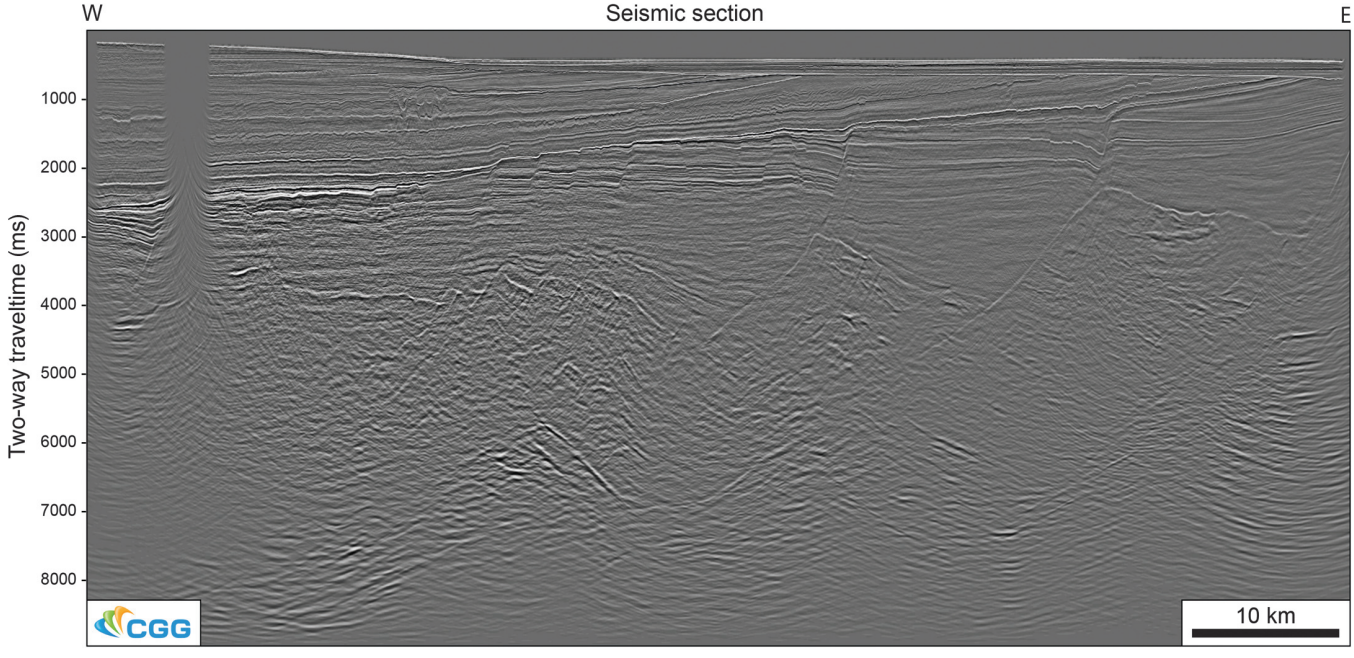

Figure 1. Seismic section (courtesy of CGG) used for automated seismic facies analysis.

schemes for seismic facies developed based on numerous studies (e.g., Brown, 2004; Bacon et al., 2007; Kearey et al., 2009). The four facies (i.e., classes) that we use for classification are: (A) continuous, horizontal reflectors, (B) continuous, dipping reflectors, (C) discontinuous, crisscrossing reflectors, and (D) discontinuous, chaotic reflectors (Figure 2). These four are probably the most common basic seismic facies. Because almost all geologic structures show at least one of these facies in seismic reflection data, classifying them accurately would allow us to map a wide range of structures.

MACHINE LEARNING

To classify seismic facies, we apply a typical machine-learning workflow (e.g., Abu-Mostafa et al., 2012). The basic idea of this workflow is to "teach" a model to identify seismic facies in a seismic section. Our workflow includes the following steps: (1) feature extraction, (2) training, (3) testing, (4) model selection, and (5) application (Figure 3).

Feature extraction

Feature extraction aims to obtain as much information as possible about the object of investigation. For this purpose, we extract so-called features, i.e., properties that describe the object we study. Here, this object is a seismic section (Figure 1) and the features are statistical properties of seismic attributes inside a moving window. Because seismic attributes have been specifically designed to highlight certain characteristics of seismic reflection data (see Randen and Sønneland, 2005; Chopra and Marfurt, 2007), they are well-suited features. After examining all seismic attributes available in Schlumberger Petrel 2015©, we extract 15 attributes that allow accurate seismic facies predictions (see Table 1; Figure 4). Seismic-attribute extraction typically involves nonlinear transformations (e.g., Hilbert transformation) of the original seismic data. As such, we can describe these calculations by:

$$\mathbf{X_i} = \mathrm{T_i}(\mathbf{X_0}). \tag{1}$$

where $\mathbf{X_0}$ is the original data, $\mathrm{T_i}$ are the transformations, and $\mathbf{X_i}$ are the resulting seismic attributes that were normalized.

Although this process provides a value at each point of the data, the nature of the seismic data requires an additional processing step. The seismic data, and therefore its attributes, contain numerous small-scale variations, which only in combination form a seismic facies. This phenomenon is captured by calculating a series of statistics inside a moving window (100×100 samples) from these attributes. These statistics are the features that we use for machine learning. Mathematically, we can describe this process as a deconstruction of the seismic attribute matrices ($\mathbf{X_i}$) into a large number of matrices ($\mathbf{X_{ij}}$) for each window:

$$\mathbf{X_i} \rightarrow \mathbf{X_{i0}}, \mathbf{X_{i1}}, \ldots \tag{2}$$

In each window, we calculate a series of statistics, i.e., the features (f_{ij}):

$$f_{ij} = f(\mathbf{X_{ij}}) = (p_{20}(\mathbf{X_{ij}}), p_{80}(\mathbf{X_{ij}}), \ldots). \tag{3}$$

The statistics that we use include the (1) 20th percentile, (2) 80th percentile, (3) mean, (4) standard deviation, (5) standard error of the mean, (6) skewness, and (7) kurtosis.

Regularization

Using a large number of features can result in overfitting whereby an overly complex model describes random errors or noise in the data. To avoid overfitting, we regularize our models, when possible, during training. Training during machine learning usually involves the minimization of the in-sample error, i.e., the difference between the predicted ($f(\mathbf{X})$) and the actual (y) result:

$$\min_f \mathbf{y} - f(\mathbf{X}). \tag{4}$$

Regularization introduces an additional constraint on the set of models:

$$\min_f \mathbf{y} - f(\mathbf{X}) + \lambda \cdot R(f). \tag{5}$$

where λ is the regularization parameter and $R(f)$ is the penalty function. The regularization parameter was selected based on a trade-off

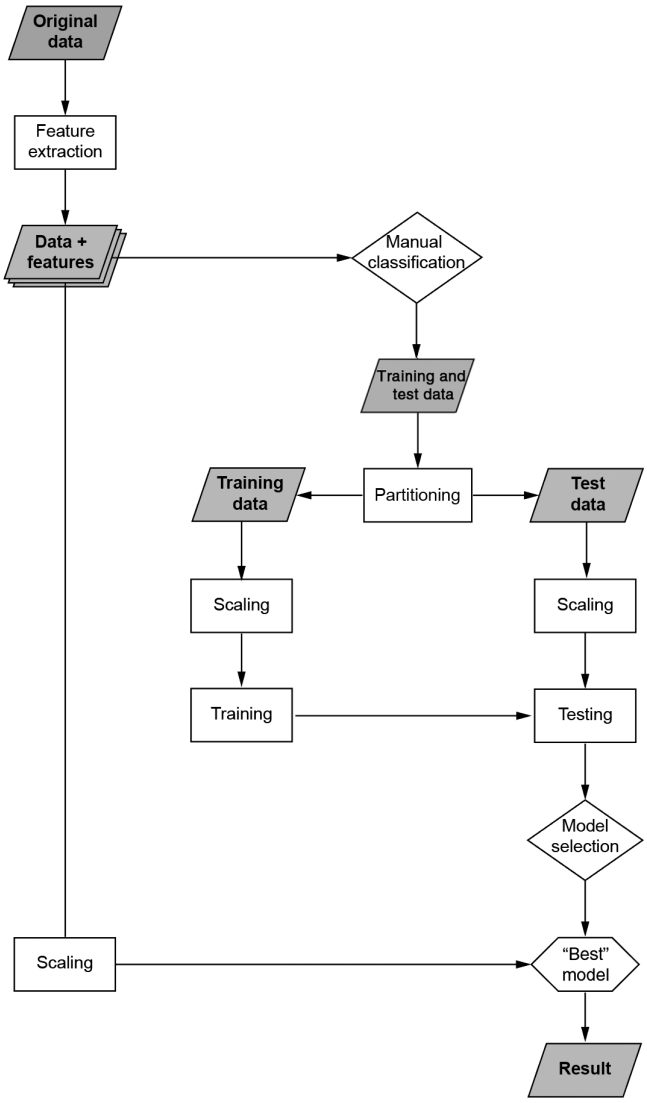

Figure 3. Machine-learning workflow in this study.

between model accuracy and simplicity during training (see Figure 5). Although we conduct no explicit feature selection, regularization can be regarded as an implicit method to constrain features.

Training

In this phase, we train 20 models to classify seismic facies using training data of our manual interpretation. Training itself involves the minimization of the in-sample error, i.e., the difference between the predicted ($f(\mathbf{X})$) and the known result ($\mathbf{y}$) (see equation 4). Because we distinguish between four seismic facies, we conduct a multiclass classification in which the model output comprises four discrete classes (A, B, C, and D). Although some classifiers can inherently handle multiclass problems, binary classifiers require one-versus-all or one-versus-one strategies to predict more than two classes. By covering the most-common algorithms used for multiclass classification (see Table 2), we are able to compare their performance on this data set (Figure 5).

To improve the performance, we explore different kernels for some of the algorithms (see Table 2). Classification problems often become easier when we transform a feature ($\mathbf{x} \in \mathbb{R}^n$) into a high-dimensional space ($\varphi(\mathbf{x}) \in \mathbb{R}^m$). However, explicit feature transformations ($\varphi: \mathbb{R}^n \rightarrow \mathbb{R}^m$) can be computationally expensive. Kernel functions (K) allow an implicit use of these high-dimensional spaces by calculating inner products between feature pair images:

$$K(\mathbf{x}, \mathbf{x}') = \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle. \tag{6}$$

As such, kernels allow us to use high-dimensional feature spaces without specifying them or the explicit transformation. Here, we use polynomial (K_{pol}) and radial basis kernel functions (K_{rbf}):

$$K_{\text{pol}}(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^{\mathbf{T}}\mathbf{x}' + c)^d, \tag{7}$$

$$K_{\text{rbf}}(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2), \tag{8}$$

in combination with support vector machines, Gaussian process, and neural network classifiers (see Table 2).

Cross-validation

During validation, we determine the model performances on yet unseen data. A simple holdout validation splits the data ($\mathbf{X}$) in two subsets: one for training ($\mathbf{X}_{\text{train}}$) and one for testing ($\mathbf{X}_{\text{test}}$). This approach, however, leads to a dilemma because we would like to maximize both subsets: the training set to generate well-constrained models and the test set to obtain reliable estimates of model performance. This dilemma is resolved by cross-validation, i.e., splitting the data multiple times and averaging performance estimates between folds. We apply a tenfold stratified cross-validation in which data are split into training (90% of the data) and test set (10% of the data) 10 times while preserving the percentage of examples of each class. To visualize model performances, we calculate an average confusion matrix of each model (Figure 6).

To quantify the model performance, we calculate (1) precision, (2) recall, and (3) f1-score of each class, and their averages for each model (see Table 3). Precision describes the ability of classifiers to predict classes correctly, recall (or sensitivity) describes the ability of classifiers to find all examples of a class, f1 score is an equally

Table 1. Extracted seismic attributes. These attributes were selected because they provide sufficient information on different geologic and geophysical characteristics of the data to allow accurate seismic facies predictions.

Seismic attributes	Highlights	Parameters
Consistent dip	Reflector dip	Output type: dip and azimuth
		Lateral filter radius: 0
		Vertical filter radius: 2
		Accuracy: 2
		AGC length: 25
Cosine of phase[3]	Structural delineations	Window length: 51
Dominant frequency[3]	Frequency content	Window length: 51
Envelope[3]	Bright spots or strong interfaces	Window length: 51
GLCM(I)[4]	Continuity	Algorithm: energy
		Lower amplitude limit: 0.0
		Upper amplitude limit: 1.0
		Levels: 5
		Split: 4
		Lateral radius: 4
		Vertical radius: 4
Instantaneous bandwidth[3]	Frequency range	Window length: 51
Instantaneous frequency[3]	Hydrocarbons, fractures, or interfaces	Window length: 51
Instantaneous phase[3]	Continuities, faults, terminations, or interfaces	Window length: 51
Instantaneous quality[3]	Fluid content or fractures	Window length: 51
Local flatnes[3]	Channels or fractures	Orientation sigma X-Z: 2.0
		Variance sigma X-Z: 2.5
Local structural dip[5]	Dip	Principal component
		Sigma X-Z: 1.5
Maximum curvature	Discontinuities or distortions	Vertical radius: 12
		Inline/crossline radius: 1
Reflection intensity	Impedance contrasts	—
Second derivative[6]	Continuity	—
Variance (edge method)	Faults and fractures	Inline range: 5
		Crossline range: 5
		Vertical smoothing:10
		Dip correction: on
		Inline scale: 1.5
		Crossline scale: 1.5
		Vertical scale: 1.5
		Plane confidence threshold: 0.6
		Dip guided smoothing: on

[3]Taner and Sheriff (1977) and Taner et al. (1979).
[4]Haralick et al. (1973), Reed and Hussong (1989), and Gao (2003).
[5]Randen et al. (2000).
[6]The second derivative attribute was calculated from the original seismic data.

DOI:10.1190/geo2017-0595.1

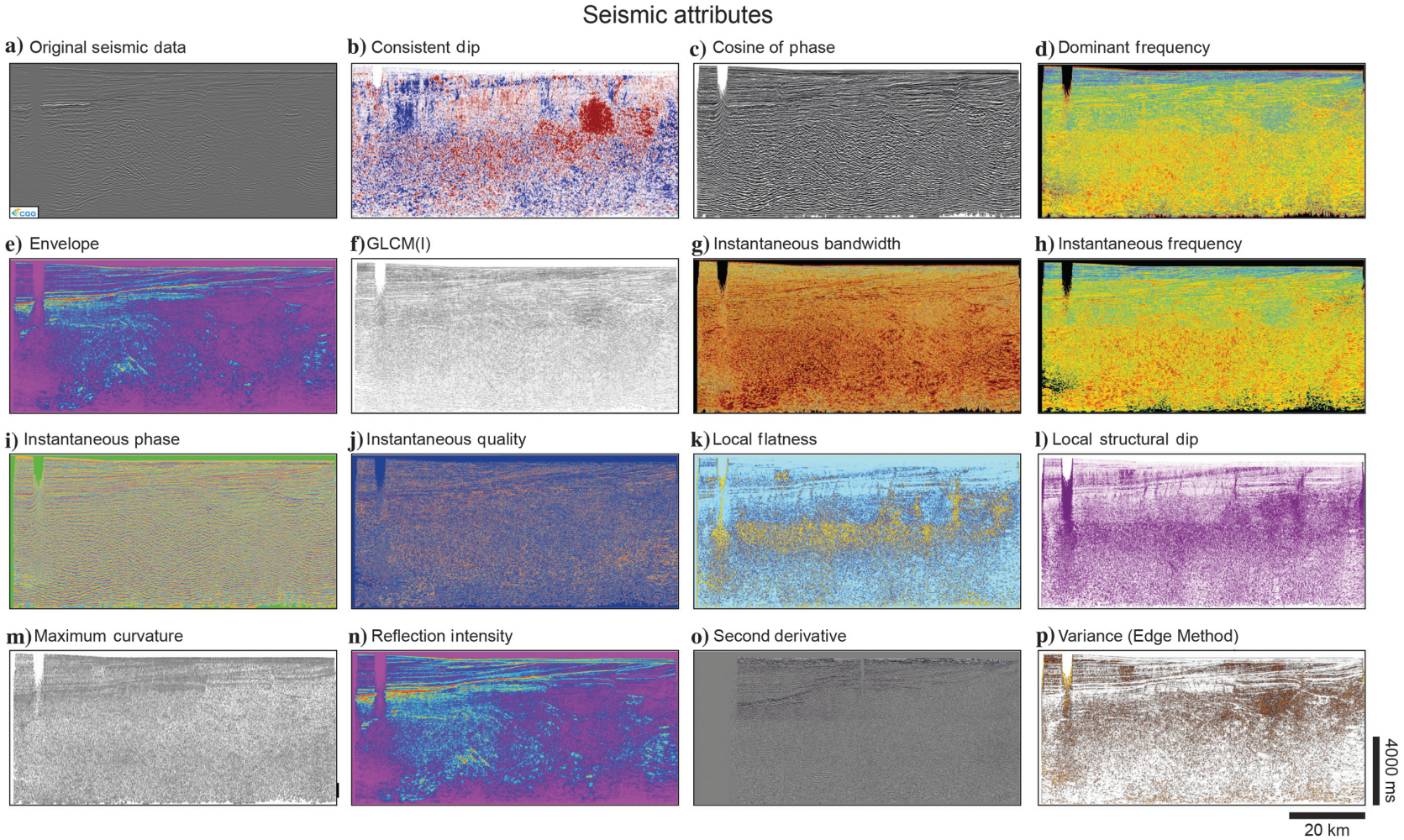

Figure 4. Calculated seismic attributes sorted in rows according to Table 1 starting with the original seismic section in the top-left corner. Seismic data courtesy of CGG.

DOI:10.1190/geo2017-0595.1

weighted harmonic mean of precision and recall, and support is the number of examples of each class. Furthermore, we calculate the average accuracy of each model and determine its standard deviation between folds (see Table 3). Regularization, training, and cross-validation were implemented in Python using the scikit-learn package (Pedregosa et al., 2011).

Model selection

Model selection is based on generalization performance of trained models on test data. In our case, the model using a support vector machine with a cubic kernel function shows the highest accuracy, precision, and recall out of all models (see Table 3; Figure 7). This means that this model not only does classify seismic facies most accurately, but it is also the best at avoiding incorrect classifications (Figure 6).

Application

After the model selection, it is recommended to train the best model again using the entire data set available, i.e., training plus test data (Abu-Mostafa et al., 2012). This final model is subsequently applied to the entire seismic section (Figure 7).

CLASSIFICATION RESULTS

Test

Test results obtained during cross-validation include confusion matrices and descriptive metrics. Confusion matrices visualize the precision of models for each class (Figure 6). When the model predicts the correct class, the sample contributes to the diagonal of the confusion matrix. When the model predicts the wrong class, the sample contributes to the cells off diagonal. The first element of the confusion matrix (top left) shows the precision for the first class (i.e., horizontal), the second element (first row, second column) shows the percentage of samples classified into the second class (i.e., dipping) despite belonging to the first class (i.e., horizontal), and so on. As such, confusion matrices show how well each model predicts each class. In general, the observed variations between models and classes are minor (<0.1) with the exception of the Gaussian process (cubic) model (Figure 6d).

To quantify these differences, we calculate a set of metrics for each model and each class (see Table 3). On the class level, these metrics include (1) precision (i.e., ability of correct classification), (2) recall (i.e., ability of complete classification), and (3) f1 score (i.e., harmonic mean of the two former). These metrics are lowest

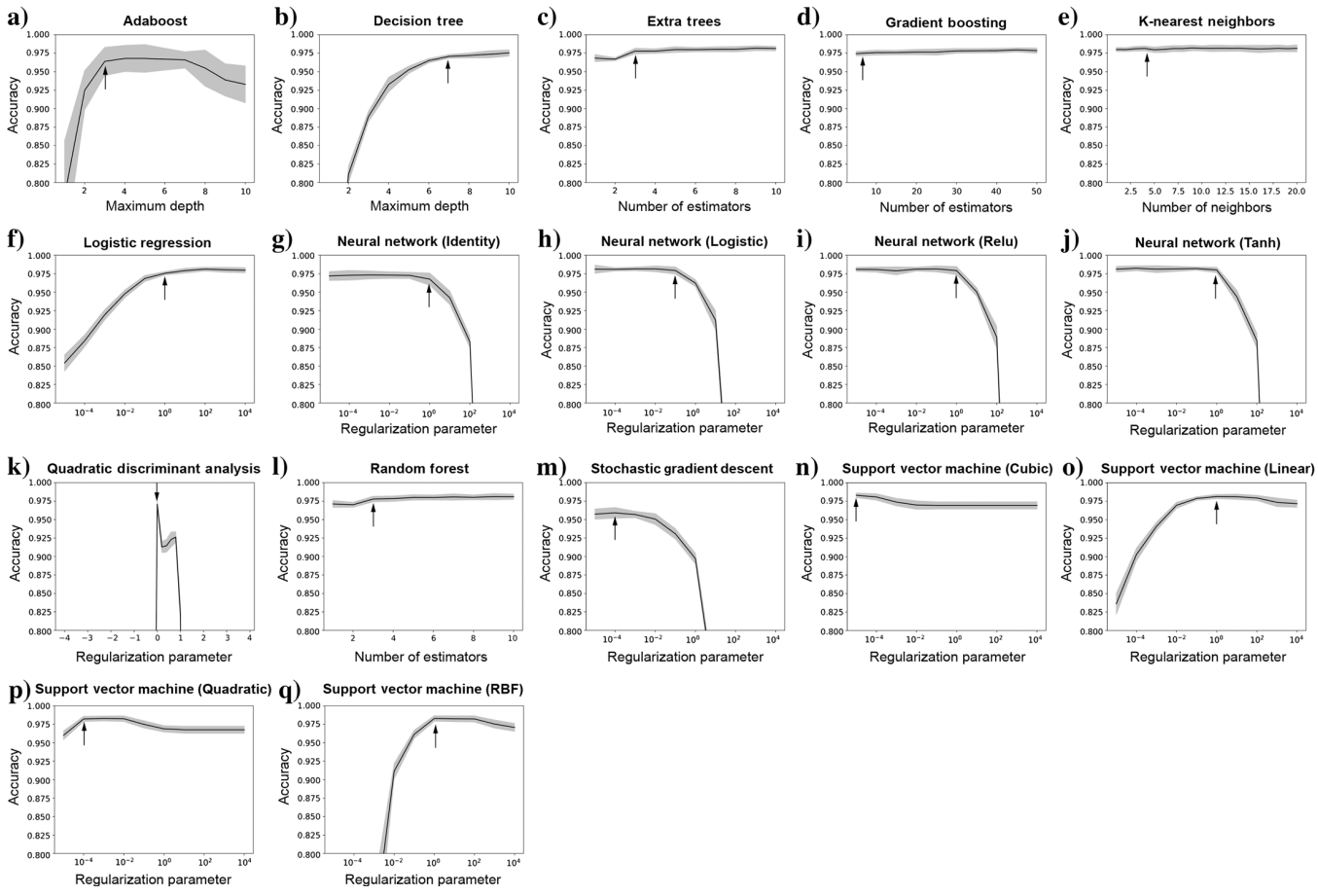

Figure 5. Hyperparameters selected based on a trade-off between model accuracy and simplicity for different algorithms during training. The gray areas indicate standard deviation of accuracy between different folds (i.e., splits) during the cross-validation.

DOI:10.1190/geo2017-0595.1

Table 2. Algorithms, default and hyperparameters in this study. The selected parameters were selected based on a trade-off between model accuracy and simplicity (see Figure 5). For more information, we refer the reader to the scikit-learn package (Pedregosa et al., 2011).

	Default settings			Hyperparameters		
Algorithm	Strategy	Kernel	Others	Name	Range	Selection
Adaboost	Multiclass	—	base_estimator = None, n_estimators = 50, learning_rate = 1.0, algorithm = 'SAMME.R', random_state = None	Max. depth	[1, 10]	3
Decision tree	Multiclass	—	criterion = 'gini', splitter = 'best', max_depth = None, min_samples_split = 2, min_samples_leaf = 1, min_weight_fraction_leaf = 0.0, max_features = None, random_state = None, max_leaf_nodes = None, min_impurity_decrease = 0.0, min_impurity_split = None, class_weight = None, presort = False	Max. depth	[1, 10]	7
Extra trees	Multiclass	—	n_estimators = 10, criterion = 'gini', max_depth = None, min_samples_split = 2, min_samples_leaf = 1, min_weight_fraction_leaf = 0.0, max_features = 'auto', max_leaf_nodes = None, min_impurity_decrease = 0.0, min_impurity_split = None, bootstrap = False, oob_score = False, n_jobs = 1, random_state = None, verbose = 0, warm_start = False, class_weight = None	Estimators	[1, 20]	3
Gaussian process (cubic kernel)	One-versus-all	Cubic polynomial	alpha = 1e-10, optimizer = 'fmin_l_bfgs_b', n_restarts_optimizer = 0, normalize_y = False, copy_X_train = True, random_state = None.	—	—	—
Gaussian process (radial basis function)	One-versus-all	Radial basis function	alpha = 1e-10, optimizer = 'fmin_l_bfgs_b', n_restarts_optimizer = 0, normalize_y = False, copy_X_train = True, random_state = None	—	—	—
Gradient boosting	Multiclass	—	loss = 'deviance', learning_rate = 0.1, n_estimators = 100, subsample = 1.0, criterion = 'friedman_mse', min_samples_split = 2, min_samples_leaf = 1, min_weight_fraction_leaf = 0.0, max_depth = 3, min_impurity_decrease = 0.0, min_impurity_split = None, init = None, random_state = None, max_features = None, verbose = 0, max_leaf_nodes = None, warm_start = False, presort = 'auto'	Estimators	[1, 40]	4
K-nearest neighbor	One-versus-all	—	radius = 1.0, algorithm = 'auto', leaf_size = 30, metric = 'minkowski', p = 2, metric_params = None, n_jobs = 1, **kwargs	Neighbors	[1, 20]	4
Linear discriminant analysis	Multiclass	—	solver = 'svd', shrinkage = None, priors = None, n_components = None, store_covariance = False, tol = 0.0001	—	—	—
Logistical regression	One-versus-all	—	penalty = 'l2', dual = False, tol = 0.0001, fit_intercept = True, intercept_scaling = 1, class_weight = None, random_state = None, solver = 'liblinear', max_iter = 100, multi_class = 'ovr', verbose = 0, warm_start = False, n_jobs = 1	Regularization	[10e-5, 10e5]	1
Neural network (identity)	One-versus-all	—	hidden_layer_sizes = (100,), activation = 'relu', solver = 'adam', batch_size = 'auto', learning_rate = 'constant', learning_rate_init = 0.001, power_t = 0.5, max_iter = 200, shuffle = True, random_state = None, tol = 0.0001, verbose = False, warm_start = False, momentum = 0.9, nesterovs_momentum = True, early_stopping = False, validation_fraction = 0.1, beta_1 = 0.9, beta_2 = 0.999, epsilon = 1e-08	Regularization	[10e-5, 10e5]	1
Neural network (logistic)	One-versus-all	—	hidden_layer_sizes = (100,), activation = 'relu', solver = 'adam', batch_size = 'auto', learning_rate = 'constant', learning_rate_init = 0.001, power_t = 0.5, max_iter = 200, shuffle = True, random_state = None, tol = 0.0001, verbose = False, warm_start = False, momentum = 0.9, nesterovs_momentum = True, early_stopping = False, validation_fraction = 0.1, beta_1 = 0.9, beta_2 = 0.999, epsilon = 1e-08	Regularization	[10e-5, 10e5]	0.1

DOI:10.1190/geo2017-0595.1

Table 2. Algorithms, default and hyperparameters in this study. The selected parameters were selected based on a trade-off between model accuracy and simplicity (see Figure 5). For more information, we refer the reader to the scikit-learn package (Pedregosa et al., 2011). (continued)

	Default settings			Hyperparameters		
Algorithm	Strategy	Kernel	Others	Name	Range	Selection
Neural network (rectified linear unit)	One-versus-all	—	hidden_layer_sizes = (100,), activation = 'relu', solver = 'adam', batch_size = 'auto', learning_rate = 'constant', learning_rate_init = 0.001, power_t = 0.5, max_iter = 200, shuffle = True, random_state = None, tol = 0.0001, verbose = False, warm_start = False, momentum = 0.9, nesterovs_momentum = True, early_stopping = False, validation_fraction = 0.1, beta_1 = 0.9, beta_2 = 0.999, epsilon = 1e-08	Regularization	[10e-5, 10e5]	1
Neural network (hyperbolic tangent)	One-versus-all	—	hidden_layer_sizes = (100,), activation = 'relu', solver = 'adam', batch_size = 'auto', learning_rate = 'constant', learning_rate_init = 0.001, power_t = 0.5, max_iter = 200, shuffle = True, random_state = None, tol = 0.0001, verbose = False, warm_start = False, momentum = 0.9, nesterovs_momentum = True, early_stopping = False, validation_fraction = 0.1, beta_1 = 0.9, beta_2 = 0.999, epsilon = 1e-08	Regularization	[10e-5, 10e5]	1
Quadratic discriminant analysis	One-versus-all	—	priors = None, store_covariance = False, tol = 0.0001, store_covariances = None	Regularization	[-2, 2]	0
Random forest	Multiclass	—	criterion = 'gini', max_depth = None, min_samples_split = 2, min_samples_leaf = 1, min_weight_fraction_leaf = 0.0, max_features = 'auto', max_leaf_nodes = None, min_impurity_decrease = 0.0, min_impurity_split = None, bootstrap = True, oob_score = False, n_jobs = 1, random_state = None, verbose = 0, warm_start = False, class_weight = None	Estimators	[1, 50]	3
Stochastic gradient descent	One-versus-all	—	loss = 'hinge', penalty = 'l2', l1_ratio = 0.15, fit_intercept = True, max_iter = None, tol = None, shuffle = True, verbose = 0, epsilon = 0.1, n_jobs = 1, random_state = None, learning_rate = 'optimal', eta0 = 0.0, power_t = 0.5, class_weight = None, warm_start = False, average = False, n_iter = None	Regularization	[10e-5, 10e5]	10e-4
Support vector machine (cubic)	One-versus-one	Cubic polynomial	degree = 3, gamma = 'auto', coef0 = 0.0, shrinking = True, probability = False, tol = 0.001, cache_size = 200, class_weight = None, verbose = False, max_iter = -1, decision_function_shape = 'ovr', random_state = None	Regularization	[10e-5, 10e5]	10e-5
Support vector machine (linear)	One-versus-one	Linear function	degree = 3, gamma = 'auto', coef0 = 0.0, shrinking = True, probability = False, tol = 0.001, cache_size = 200, class_weight = None, verbose = False, max_iter = -1, decision_function_shape = 'ovr', random_state = None	Regularization	[10e-5, 10e5]	1
Support vector machine (quadratic)	One-versus-one	Quadratic polynomial	degree = 3, gamma = 'auto', coef0 = 0.0, shrinking = True, probability = False, tol = 0.001, cache_size = 200, class_weight = None, verbose = False, max_iter = -1, decision_function_shape = 'ovr', random_state = None	Regularization	[10e-5, 10e5]	10e-4
Support vector machine (radial basis function)	One-versus-one	Radial basis function	degree = 3, gamma = 'auto', coef0 = 0.0, shrinking = True, probability = False, tol = 0.001, cache_size = 200, class_weight = None, verbose = False, max_iter = -1, decision_function_shape = 'ovr', random_state = None	Regularization	[10e-5, 10e5]	1

for Facies B (i.e., continuous, dipping reflections) and Facies D (i.e., discontinuous, chaotic reflections) for almost all models (Table 3). To assess the overall model performance, we calculate averages of these metrics for each model as well as the accuracy and its standard deviations between folds for each model. Table 3 shows all trained models sorted from highest (0.983) to lowest

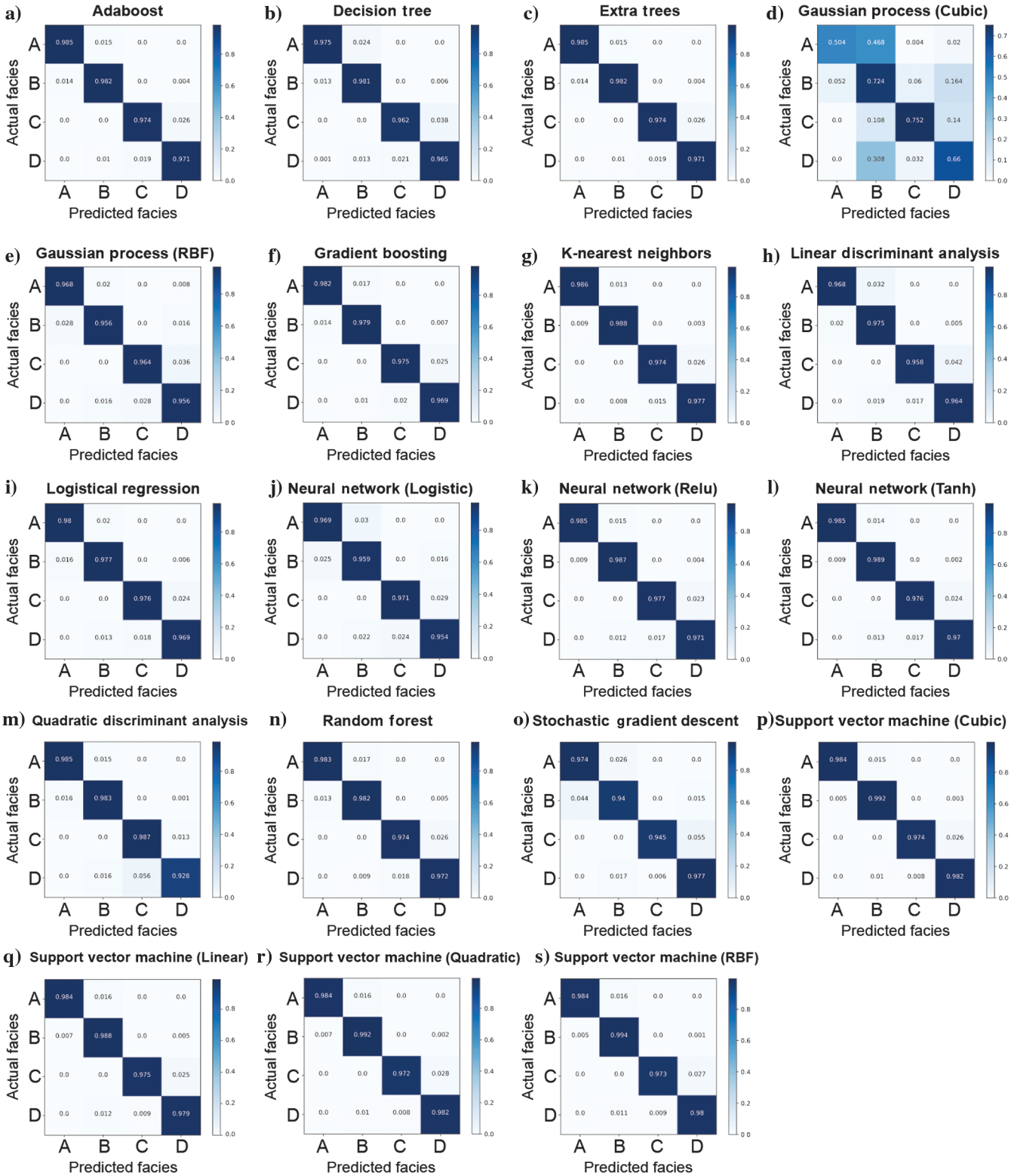

Figure 6. Confusion matrices of trained models showing average number of correct classifications on diagonal and average number of incorrect classifications off diagonal. Averages result from tenfold cross-validation. Note that the classes were balanced, so that the confusion matrices visualize class-wise model accuracies.

DOI:10.1190/geo2017-0595.1

DOI:10.1190/geo2017-0595.1

Table 3 . Parameters describing model performance on each class (i.e., facies) and overall, i.e., precision, recall, f1-score, and support as well as accuracy. Models are sorted by accuracy. Standard deviations of model accuracies between different folds of cross-validation are listed in the last column. Note that models are balanced with a support of 2500 for each class.

		Facies A			Facies B			Facies C			Facies D			Overall			
No.	Model	Pre	Rec	f1	Pre	Rec	f1	Pre	Rec	f1	Pre	Rec	f1	Pre	Rec	f1	Acc
1	Support vector machine (cubic)	0.995	0.985	0.990	0.975	0.992	0.983	0.992	0.974	0.983	0.971	0.982	0.977	0.983	0.983	0.983	0.983 ± 0.004
2	Support vector machine (radial basis function)	0.995	0.984	0.989	0.974	0.994	0.984	0.991	0.973	0.982	0.973	0.980	0.977	0.983	0.983	0.983	0.983 ± 0.004
3	Support vector machine (quadratic)	0.993	0.984	0.989	0.974	0.992	0.983	0.992	0.972	0.982	0.971	0.982	0.976	0.983	0.982	0.982	0.983 ± 0.004
4	Support vector machine (linear)	0.993	0.984	0.988	0.973	0.988	0.980	0.991	0.975	0.983	0.970	0.979	0.974	0.982	0.981	0.981	0.982 ± 0.004
5	K-nearest neighbor	0.991	0.987	0.989	0.979	0.988	0.983	0.985	0.974	0.979	0.972	0.977	0.974	0.982	0.981	0.981	0.982 ± 0.003
6	Neural network (rectified linear unit)	0.994	0.982	0.988	0.966	0.992	0.979	0.988	0.976	0.982	0.975	0.970	0.972	0.980	0.980	0.980	0.980 ± 0.005
7	Neural network (logistic activation function)	0.990	0.984	0.987	0.972	0.982	0.977	0.988	0.975	0.982	0.968	0.975	0.971	0.980	0.979	0.979	0.980 ± 0.004
8	Extra tree	0.986	0.985	0.986	0.975	0.982	0.979	0.981	0.974	0.977	0.969	0.971	0.970	0.978	0.978	0.978	0.978 ± 0.004
9	Random forest	0.987	0.983	0.985	0.974	0.982	0.978	0.982	0.974	0.977	0.969	0.972	0.971	0.978	0.978	0.978	0.978 ± 0.004
10	Neural network (hyperbolic tangent activation function)	0.990	0.984	0.987	0.969	0.983	0.976	0.983	0.976	0.979	0.970	0.967	0.968	0.978	0.977	0.977	0.978 ± 0.003
11	Gradient boosting	0.986	0.983	0.984	0.973	0.979	0.976	0.980	0.975	0.977	0.969	0.969	0.969	0.977	0.977	0.977	0.977 ± 0.003
12	Logistic regression	0.984	0.980	0.982	0.968	0.977	0.973	0.982	0.976	0.979	0.970	0.969	0.969	0.976	0.976	0.976	0.976 ± 0.003
13	Quadratic discriminant analysis	0.984	0.985	0.985	0.970	0.983	0.976	0.947	0.987	0.966	0.985	0.928	0.956	0.971	0.971	0.971	0.971 ± 0.004
14	Decision tree	0.986	0.976	0.981	0.963	0.981	0.972	0.978	0.962	0.970	0.957	0.965	0.961	0.971	0.971	0.971	0.971 ± 0.003
15	Neural network (identity activation function)	0.971	0.973	0.972	0.957	0.962	0.959	0.983	0.976	0.979	0.968	0.966	0.967	0.970	0.969	0.969	0.970 ± 0.005
16	Adaboost	0.977	0.972	0.974	0.966	0.972	0.968	0.972	0.964	0.968	0.961	0.964	0.962	0.969	0.968	0.968	0.969 ± 0.011
17	Linear discriminant analysis	0.980	0.968	0.974	0.951	0.975	0.963	0.983	0.958	0.970	0.954	0.964	0.959	0.967	0.967	0.967	0.967 ± 0.004
18	Gaussian process (cubic)	0.973	0.972	0.972	0.965	0.956	0.960	0.973	0.964	0.968	0.943	0.956	0.948	0.964	0.962	0.962	0.964 ± 0.004
19	Stochastic gradient descent	0.957	0.974	0.965	0.957	0.940	0.948	0.993	0.945	0.969	0.934	0.977	0.955	0.960	0.959	0.959	0.960 ± 0.004
20	Gaussian process (radial basis function)	0.762	0.504	0.525	0.587	0.724	0.628	0.889	0.752	0.789	0.701	0.660	0.647	0.735	0.660	0.647	0.735 ± 0.127

accuracy (0.735). These results are consistent as the model with the highest accuracy (support vector machine (cubic)) also shows the highest precision, recall, and f1-score.

Application

Applying the best model (support vector machine (cubic)) to the entire seismic section produces the final results shown on Figure 7. The results show that the model is, to first degree, able to identify the seismic facies that we use for training: (A) continuous, horizontal reflectors; (B) continuous, dipping reflectors; (C) discontinuous, criss-crossing reflectors; and (D) discontinuous, chaotic reflectors in the seismic section. In general, the model is able to distinguish the sedimentary succession (Facies A and B) from the basement (Facies C and D). Within the sedimentary succession, the model is also able to identify some details, such as (1) the vertical artifact on the left side of the section, (2) dipping reflectors associated with the artifact, (3) clastic remobilizations, and (4) fault-propagation folds of the major tectonic faults. Within the basement, the model succeeds in identifying areas in which strong reflectors crisscross each other (Facies C).

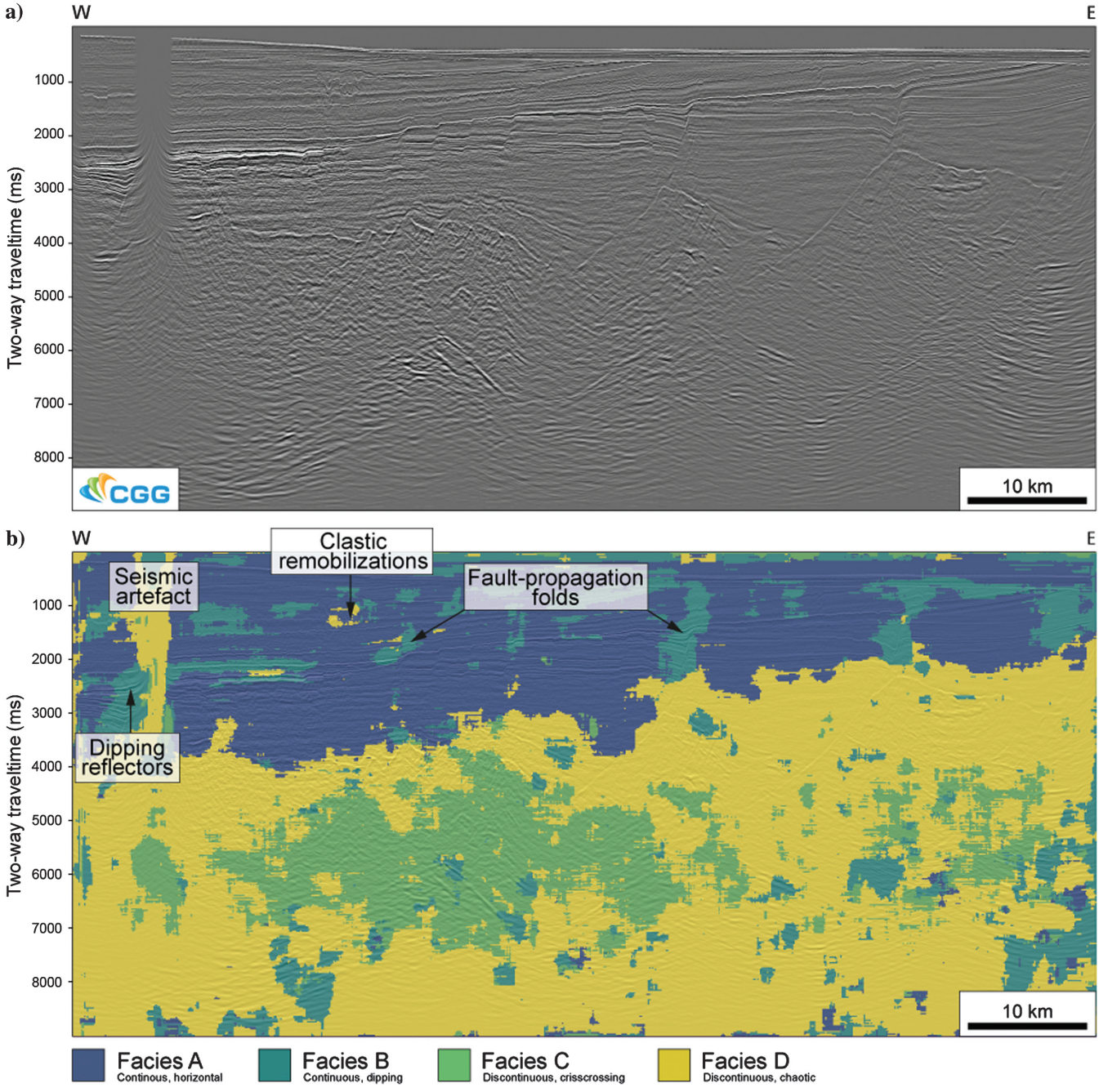

Figure 7. (a) Seismic section and (b) final classification result. Seismic data courtesy of CGG.

DOI:10.1190/geo2017-0595.1

DOI:10.1190/geo2017-0595.1

DISCUSSION

This study demonstrates the applicability of machine-learning techniques to seismic facies analyses. By applying a supervised machine-learning workflow to state-of-the-art seismic reflection data, we are able to classify the main seismic facies in a seismic section (Figure 7). In contrast to previous studies, which often analyze facies at each point of a data set (see the review by Zhao et al., 2015), our classification is based on a window around each point. This process is equivalent to a deconstruction of the data set into a large number of matrices, each corresponding to one point in the data set. This approach mimics manual facies interpretations, which consider patterns within a certain area, rather than at a specific point. Although this approach might not resolve all details, it is probably more robust to noise.

Although previous studies focused on unsupervised machine-learning (e.g., Coléou et al., 2003; de Matos et al., 2006), this study applies supervised learning, i.e., a classification based on predefined seismic facies. This approach has several advantages when it comes to the final geologic interpretation. The seismic facies derived by unsupervised learning can often be difficult to associate with existing geologic units or structures. In fact, the nature of these automatically extracted facies can remain completely unknown to the interpreter, introducing ambiguities and doubts into the interpretation. However, supervised learning allows us to decide on the number and types of seismic facies to include. This allows us to adapt our analysis to map a wide range of geologic units and structures, such as salt, sand, and magmatic bodies as well as folds, faults, and shear zones.

One may now argue that the supervised learning approach reintroduces an element of subjectivity to the analysis. However, this argument presupposes that unsupervised learning is entirely objective. This is not the case. Even unsupervised learning requires the selection of certain criteria (e.g., the number of classes, choice of objective function). One may thus argue that supervised learning increases the degree of subjectivity in the analysis. Although this appears to be the case, it is worth noting that subjectivity is not a problem per se — in many cases, it is desirable to include expert input as long as the applied workflow remains repeatable. As long as other researchers are able to replicate a workflow while reaching the same results, we maintain repeatability. This is a significant improvement over conventional seismic interpretations, in which multiple researchers can reach different conclusions applying seemingly the same method.

Another advantage of machine learning over conventional seismic interpretation is its ability to obtain quantitative metrics, such as model accuracy. More precisely, we are able to quantify the prediction accuracy on yet unseen data (see Table 3). This is done by cross-validation, where a data set is split several times into training and test data. Applying cross-validation in this study yields prediction accuracies ranging from 0.735 to 0.983, depending on the algorithm used. Given this information, we can (1) select the model with the highest accuracy (support vector machine (cubic)), (2) train this model using the entire data set (training plus test data), and (3) apply it to the seismic section (Figure 7).

A qualitative analysis of the final model results demonstrates that the model is able to identify the four main seismic facies (Figure 7), which implies that the model can, to a certain degree, distinguish between sedimentary (Facies A and B) and basement rocks (Facies C and D). Within the sedimentary succession, the model is even able to identify some details, such as (1) the vertical artifact on the left side of the section, (2) dipping reflectors associated with the artifact, (3) clastic remobilizations, and (4) fault-propagation folds of the major tectonic faults. This suggests that the algorithm could be further optimized to detect any of these geologic or geophysical features. Within the basement, the model succeeds in identifying areas where strong reflectors crisscross each other (Facies C) — a signature that has been used by conventional seismic interpretations of basement rocks (e.g., Phillips et al., 2016; Fazlikhani et al., 2017). Future work can focus on mapping seismic facies within the basement to a greater level of detail, identifying key rock units and structures in the northern North Sea (e.g., the Western Gneiss Region and the Hardangerfjord Shear Zone).

Finally, it may be argued that an experienced seismic interpreter can obtain a much more detailed seismic interpretation and, thus, a deeper geologic understanding of the seismic section above than our best model. First, it is worth remembering that a greater level of detail does not imply that the interpretation is necessarily correct. Seismic interpreters are prone to project geologic structures that they are familiar with onto new data (Bond et al., 2007). Second, our approach could also be adapted to highlight additional geologic details. This would only require the definition of additional seismic facies. Third, the aim of this study is to support, rather than compete with, conventional seismic interpretations. We believe that machine learning can become a valuable tool to seismic interpreters aiming to raise prediction accuracy and to reduce uncertainty.

CONCLUSION

We demonstrate the applicability of supervised machine learning to seismic facies analyses. The basis of this study is state-of-the-art broadband 3D seismic reflection data of the northern North Sea rift, to which we apply a typical machine-learning workflow including (1) feature extraction, (2) training, (3) testing, (4) model selection, and (5) application. This workflow allows us to generate models that predict seismic facies with accuracies of up to 0.983 $\pm$ 0.004. The model with the highest accuracy uses a regularized support vector machine to predict seismic facies. Applying this model to an entire seismic section demonstrates that it is able to provide an effective seismic facies analysis. This highlights that machine-learning has the potential to change the way we analyze seismic reflection data in the future.

ACKNOWLEDGMENTS

First, we would like to thank the journal editors (Deyan Draganov, Valentina Socco and Sergio Chávez-Pérez) and the reviewers (Brendan Hall, Rocky Roden and two anonymous reviewers). We also thank The Norwegian Academy of Science and Letters (VISTA) and The University of Bergen for supporting this research. We are very grateful to CGG for supplying seismic data and allowing us to publish this work. In particular, the support of Stein Åsheim and Marit Stokke Bauck is greatly appreciated. Schlumberger is thanked for providing the software Petrel 2015©. We thank the developers of python and scikit-learn, which was used to implement this workflow and we thank Leo Zijerveld for IT support. Finally, we would like to thank the members of the MultiRift project, in particular Antje Lenhart and Tom Phillips, for numerous discussions leading to the development of this study.

REFERENCES

Abu-Mostafa, Y. S., M. Magdon-Ismail, and H.-T. Lin, 2012, Learning from data: AML Book: AML Book .

Araya-Polo, M., T. Dahlke, C. Frogner, C. Zhang, T. Poggio, and D. Hohl, 2017, Automated fault detection without seismic processing: The Leading Edge, **36**, 208–214, doi: 10.1190/tle36030208.1.

Bacon, M., R. Simm, and T. Redshaw, 2007, 3-D seismic interpretation: Cambridge University Press.

Barnes, A. E., and K. J. Laughlin, 2002, Investigation of methods for unsupervised classification of seismic data: 72nd Annual International Meeting, SEG, Expanded Abstracts, 2221–2224, doi: 10.1190/1.1817152.

Baudon, C., and J. A. Cartwright, 2008, 3D seismic characterisation of an array of blind normal faults in the Levant Basin, Eastern Mediterranean: Journal of Structural Geology, **30**, 746–760, doi: 10.1016/j.jsg.2007.12.008.

Bond, C. E., 2015, Uncertainty in structural interpretation: Lessons to be learnt: Journal of Structural Geology, **74**, 185–200, doi: 10.1016/j.jsg.2015.03.003.

Bond, C. E., A. D. Gibbs, Z. K. Shipton, and S. Jones, 2007, What do you think this is? "Conceptual uncertainty" in geoscience interpretation: GSA Today, **17**, 4–10, doi: 10.1130/GSAT01711A.1.

Bond, C. E., R. Lunn, Z. Shipton, and A. Lunn, 2012, What makes an expert effective at interpreting seismic images?: Geology, **40**, 75–78, doi: 10.1130/G32375.1.

Brown, A. R., 2004, Interpretation of three-dimensional seismic data: AAPG and SEG.

Bull, S., J. Cartwright, and M. Huuse, 2009, A review of kinematic indicators from mass-transport complexes using 3D seismic data: Marine and Petroleum Geology, **26**, 1132–1151, doi: 10.1016/j.marpetgeo.2008.09.011.

Cartwright, J., and M. Huuse, 2005, 3D seismic technology: The geological 'Hubble': Basin Research, **17**, 1–20, doi: 10.1111/j.1365-2117.2005.00252.x.

Chopra, S., and K. J. Marfurt, 2007, Seismic attributes for prospect identification and reservoir characterization: SEG and EAGE.

Coléou, T., M. Poupon, and K. Azbel, 2003, Unsupervised seismic facies classification: A review and comparison of techniques and implementation: The Leading Edge, **22**, 942–953, doi: 10.1190/1.1623635.

de Matos, M. C., P. L. Osorio, and P. R. Johann, 2006, Unsupervised seismic facies analysis using wavelet transform and self-organizing maps: Geophysics, **72**, no. 1, P9–P21, doi: 10.1190/1.2392789.

Fazlikhani, H., H. Fossen, R. Gawthorpe, J. I. Faleide, and R. E. Bell, 2017, Basement structure and its influence on the structural configuration of the northern North Sea rift: Tectonics, **36**, 1151–1177, doi: 10.1002/2017TC004514.

Firth, J., I. Horstad, and M. Schakel, 2014, Experiencing the full bandwidth of energy from exploration to production with the art of BroadSeis: First Break, **32**, 89–97.

Gao, D., 2007, Application of three-dimensional seismic texture analysis with special reference to deep-marine facies discrimination and interpretation: Offshore Angola, west Africa: AAPG Bulletin, **91**, 1665–1683, doi: 10.1306/08020706101.

Gao, D. L., 2003, Volume texture extraction for 3D seismic visualization and interpretation: Geophysics, **68**, 1294–1302, doi: 10.1190/1.1598122.

Guillen, P., G. Larrazabal, G. González, D. Boumber, and R. Vilalta, 2015, Supervised learning to detect salt body: 85th Annual International Meeting, SEG, Expanded Abstracts, 1826–1829, doi: 10.1190/segam2015-5931401.1.

Hale, D., 2013, Methods to compute fault images, extract fault surfaces, and estimate fault throws from 3D seismic images: Geophysics, **78**, no. 2, O33–O43, doi: 10.1190/geo2012-0331.1.

Hansen, D. M., J. A. Cartwright, and D. Thomas, 2004, 3D seismic analysis of the geometry of igneous sills and sill junction relationships: Geological Society, London, Memoirs 29, 199–208.

Haralick, R. M., K. Shanmugam, and I. Dinstein, 1973, Textural features for image classification: IEEE Transactions on Systems Man and Cybernetics, **Smc3**, 610–621, doi: 10.1109/Tsmc.1973.4309314.

Huang, L., X. Dong, and T. E. Clee, 2017, A scalable deep learning platform for identifying geologic features from seismic attributes: The Leading Edge, **36**, 249–256, doi: 10.1190/tle36030249.1.

Jackson, C. A., M. P. Jackson, M. R. Hudec, and C. Rodriguez, 2014, Internal structure, kinematics, and growth of a salt wall: Insights from 3-D seismic data: Geology, **42**, 307–310, doi: 10.1130/G34865.1.

Kearey, P., M. Brooks, and I. Hill, 2009, An introduction to geophysical exploration: John Wiley & Sons.

Macrae, E. J., C. E. Bond, Z. K. Shipton, and R. J. Lunn, 2016, Increasing the quality of seismic interpretation: Interpretation, **4**, no. 3, T395–T402, doi: 10.1190/INT-2015-0218.1.

Magee, C., E. Hunt-Stewart, and C. A.-L. Jackson, 2013, Volcano growth mechanisms and the role of sub-volcanic intrusions: Insights from 2D seismic reflection data: Earth and Planetary Science Letters, **373**, 41–53, doi: 10.1016/j.epsl.2013.04.041.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, 2011, Scikit-learn: Machine Learning in Python: Journal of Machine Learning Research, **12**, 2825–2830.

Phillips, T. B., C. A. Jackson, R. E. Bell, O. B. Duffy, and H. Fossen, 2016, Reactivation of intrabasement structures during rifting: A case study from offshore southern Norway: Journal of Structural Geology, **91**, 54–73, doi: 10.1016/j.jsg.2016.08.008.

Planke, S., T. Rasmussen, S. Rey, and R. Myklebust, 2005, Seismic characteristics and distribution of volcanic intrusions and hydrothermal vent complexes in the Vøring and Møre basins: Presented at the Geological Society, London, Petroleum Geology Conference Series, doi: 10.1144/0060833.

Posamentier, H. W., 2004, Seismic geomorphology: Imaging elements of depositional systems from shelf to deep basin using 3D seismic data: Implications for exploration and development: Geological Society, London, Memoirs 29, 11–24.

Posamentier, H. W., and V. Kolla, 2003, Seismic geomorphology and stratigraphy of depositional elements in deep-water settings: Journal of Sedimentary Research, **73**, 367–388, doi: 10.1306/111302730367.

Randen, T., E. Monsen, C. Signer, A. Abrahamsen, J. O. Hansen, T. Sæter, and J. Schlaf, 2000, Three-dimensional texture attributes for seismic data analysis: 70th Annual International Meeting, SEG, Expanded Abstracts, 668–671, doi: 10.1190/1.1816155.

Randen, T., and L. Sønneland, 2005, Atlas of 3D seismic attributes, Mathematical methods and modelling in hydrocarbon exploration and production: Springer, 23–46.

Reed, T. B., and D. Hussong, 1989, Digital image-processing techniques for enhancement and classification of Seamarc-Ii side scan sonar imagery: Journal of Geophysical Research-Solid Earth and Planets, **94**, 7469–7490, doi: 10.1029/JB094iB06p07469.

Stewart, S. A., and P. J. Allen, 2002, A 20-km diameter multi-ringed impact structure in the North Sea: Nature, **418**, 520–523, doi: 10.1038/nature00914.

Taner, M., and R. E. Sheriff, 1977, Application of amplitude, frequency, and other attributes to stratigraphic and hydrocarbon determination: Section 2. Application of seismic reflection configuration to stratigraphic interpretation: AAPG.

Taner, M. T., F. Koehler, and R. E. Sheriff, 1979, Complex seismic trace analysis: Geophysics, **44**, 1041–1063, doi: 10.1190/1.1440994.

Underhill, J. R., 2004, Earth science — An alternative origin for the 'Silverpit crater': Nature, **428**, 1–2, doi: 10.1038/nature02476.

Vail, P. R., 1987, Seismic stratigraphy interpretation using sequence stratigraphy — Part 1: Seismic stratigraphy interpretation procedure: AAPG Special Volumes, **27**, 1–10.

Zhang, C., C. Frogner, M. Araya-Polo, and D. Hohl, 2014, Machine-learning based automated fault detection in seismic traces: 76th Annual International Conference and Exhibition, EAGE, Extended Abstracts, doi: 10.3997/2214-4609.20141500.

Zhao, T., V. Jayaram, A. Roy, and K. J. Marfurt, 2015, A comparison of classification techniques for seismic facies recognition: Interpretation, **3**, no. 4, SAE29–SAE58, doi: 10.1190/INT-2015-0044.1.