A comparison of classification techniques for seismic facies recognition

Tao Zhao¹, Vikram Jayaram², Atish Roy³, and Kurt J. Marfurt¹

¹The University of Oklahoma, ConocoPhillips School of Geology and Geophysics

²Oklahoma Geological Survey

³BP America

tao-zhao@ou.edu, vjayaram@ou.com, atish.roy@bp.com, and kmarfurt@ou.edu

Corresponding author:

Tao Zhao

The University of Oklahoma, ConocoPhillips School of Geology and Geophysics

710 Sarkeys Energy Center

100 East Boyd Street

Norman, OK 73019

tao-zhao@ou.edu
ABSTRACT

During the past decade, the size of 3D seismic data volumes and the number of seismic attributes have increased to the extent that it difficult, if not impossible, for interpreters to examine every seismic line and time slice. To address this problem, several seismic facies classification algorithms including $k$-means, self-organizing maps, generative topographic mapping, support vector machines, Gaussian mixture models, and artificial neural networks have been successfully used to extract features of geologic interest from multiple volumes. Although well documented in the literatures, the terminology and complexity of these algorithms may bewilder the average seismic interpreter, and few papers have applied these competing methods to same data volume. In this paper, we review six commonly used algorithms and apply them to a single 3D seismic data volume acquired over the Canterbury Basin, offshore New Zealand, where one of the main objectives is to differentiate the architectural elements of a turbidite system. Not surprisingly, the most important parameter in this analysis is the choice of the correct input attributes, which in turn depends on careful pattern recognition by the interpreter. We find that supervised learning methods provide accurate estimates of the desired seismic facies, while unsupervised learning methods also highlight features that may otherwise be overlooked.

Keywords: pattern recognition, machine learning, classification, seismic interpretation, reservoir characterization
INTRODUCTION

In 2015, pattern recognition has become part of everyday life. Amazon or Alibaba analyzes the clothes you buy, Google analyzes your driving routine, and your local grocery store knows the kind of cereal you eat in the morning. “Big data” and “deep learning algorithms” are being analyzed by big companies and big government, attempting to identify patterns in our spending habits and the people with whom we associate.

Successful seismic interpreters are experts at pattern recognition, identifying features such as channels, mass transport complexes, and collapse features where our engineering colleagues only see wiggles. Our challenge as interpreters is that the data volumes we need to analyze keep growing in size and dimensionality, while the number of experienced interpreters has remained relatively constant. One solution to this dilemma is for these experienced interpreters to teach their skills to the next generation of geologists and geophysicists, either through traditional or on the job training. An alternative and complimentary solution is for these experienced interpreters to teach theirs skills to a machine. Turing (1950), whose scientific contributions and life has recently been popularized in a movie, asked whether “Machines can think?” Whether machines will ever be able to think is a question for scientists and philosophers to answer (e.g. Eagleman, 2012), but machines can be taught to perform repetitive tasks, and even to unravel the relationships that underlay repetitive patterns, in an area called “Machine Learning”.

25 years ago, skilled interpreters delineated seismic facies on a suite of 2D lines by visually examining seismic waveforms, frequency, amplitude, phase, and geometric configurations. Facies would then be posted on a map and hand contoured to generate a seismic facies map. With the introduction of 3D seismic data and volumetric attributes, such analysis has become both more
quantitative and more automated. In this tutorial, we focus on classification (also called clustering) on large 3D seismic data whereby like patterns in the seismic response (seismic facies) are assigned similar values. Much of the same technology can be used to define specific rock properties, such as brittleness, TOC, or porosity. Pattern recognition and clustering is common to many industries, from using cameras to identify knotholes in plywood production to tracking cell phone communications to identify potential narcotics traffickers. The workflow is summarized in the classic textbook by Duda et al. (2001) displayed in Figure 1. In this figure, “sensing” consists of seismic, well log, completion, and production measurements. For interpreters “segmentation” will usually mean focusing on a given stratigraphic formation or suite of formations. Seismic data lose both temporal and lateral resolution with depth, such that a given seismic facies changes its appearance, or is nonstationary, as we go deeper in the section. The number of potential facies also increases as we analyze larger vertical windows incorporating different depositional environments, making classification more difficult. For computer assisted facies classification, “feature extraction” means attributes, be they simple measurements of amplitude and frequency, geometric attributes that measure reflector configurations, or more quantitative measurements of lithology, fractures, or geomechanical properties provided by prestack inversion and azimuthal anisotropy analysis. “Classification” assigns each voxel to one of a finite number of classes (also called clusters), each of which represents a seismic facies that may or may not correspond to a geological facies. Finally, using validation data, the interpreter makes a “decision” that determines whether a given cluster represents a unique seismic facies, if it should be lumped in other clusters having a somewhat similar attribute expression, or whether it should be further subdivided, perhaps through the introduction of additional attributes.
Pattern recognition and classification of seismic features is fundamental to human based interpretation, where our job may be as “simple” as identifying and picking horizons and faults, or more advanced such as the delineation of channels, mass transport complexes, carbonate buildups, or potential gas accumulations. The use of computer-assisted classification began soon after the development of seismic attributes in the 1970s (Balch, 1971; Taner et al., 1979), with the work by Sonneland (1983) and Justice et al. (1985) being two of the first. $K$-means (Forgy, 1965; Jancey, 1966) was one of the earliest clustering algorithms developed, and was quickly applied by service companies and today is common to almost all interpretation software packages. $K$-means is an unsupervised learning algorithm in that the interpreter provides no prior information other than the selection of attributes and the number of desired clusters.

Barnes and Laughlin (2002) reviewed several unsupervised clustering techniques, including $k$-means, fuzzy clustering, and self-organizing maps (SOM). Their primary finding was that the clustering algorithm used was less important than the choice of attributes used. Among the clustering algorithms, they favored SOM since there is topologically ordered mapping of the clusters with similar clusters lying adjacent to each other on a manifold and in the associated latent space. In our examples, a “manifold” is a deformed 2D surface that best fits the distribution of $n$ attributes lying in an $n$-dimensional data space. The clusters are then mapped to a simpler 2D rectangular “latent” (Latin for “hidden”) space upon which the interpreter can either interactively define clusters or simply map the projections onto a 2D colorbar. A properly chosen latent space can help identify data properties that are otherwise difficult to observe in the original input space. Coleou et al.’s (2003) seismic “waveform classification” algorithm is implemented using SOM, where the “attributes” are seismic amplitudes that lie on a suite of 16 phantom horizon slices. Each $(x,y)$ location in the analysis window results provides a 16-dimensional vector of amplitudes. When
plotted one element after the other, the mean of each cluster in 16-dimensional space looks like a waveform. These waveforms lie along a 1D deformed string (the manifold) that lies in 16D. This 1D string is then mapped to a 1D line (the latent space) which in turn is mapped against a 1D continuous color bar. The proximity of like waveforms to each other on the manifold and latent spaces results in similar seismic facies appearing as similar colors. Coleou et al. (2003) generalized their algorithm to attributes other than seismic amplitude, constructing vectors of dip magnitude, coherence, and reflector parallelism. Strecker and Uden (2002) were perhaps the first to use 2D manifolds and 2D latent spaces with geophysical data, using multidimensional attribute volumes to form N-dimensional vectors at each seismic sample point. Typical attributes included envelope, bandwidth, impedance, AVO slope and intercept, dip magnitude, and coherence. These attributes were projected onto a 2D latent space and their results plotted against a 2D color table. Gao (2007) applied a 1D SOM to GLCM texture attributes to map seismic facies offshore Angola. Overdefining the clusters with 256 prototype vectors, he then used 3D visualization and his knowledge of the depositional environment to map the “natural” clusters. These natural clusters were then calibrated using well control, giving rise to what is called a posteriori supervision. Roy et al. (2013) built on these concepts and developed an SOM classification workflow of multiple seismic attributes computed over a deep-water depositional system. They calibrated the clusters a posteriori using classical principles of seismic stratigraphy on a subset of vertical slices through the seismic amplitude. A simple but very important innovation was to project the clusters onto a 2D nonlinear Sammon space (Sammon, 1969). This projection was then colored using a gradational 2D color-scale like that of Matos et al. (2009) thus facilitating the interpretation. Roy et al. (2013) introduced a Euclidean distance measure to correlate predefined unsupervised clusters to average data vectors about interpreter defined well log facies.
Generative topographic mapping (GTM) is a more recent unsupervised classification innovation, providing a probabilistic representation of the data-vectors in the latent space (Bishop et al., 1998). There has been very little work on the application of GTM technique to seismic data and exploration problems. Wallet et al. (2009) are probably the first to apply the GTM technique to seismic data, using a suite of phantom horizon slices through a seismic amplitude volume generating a “waveform classification”. While generating excellent images, Roy et al. (2013, 2014) found the introduction of well control to SOM classification to be somewhat limited, and instead applied generative topographic mapping (GTM) to Mississippian tripolitic chert reservoir in the Midcontinent USA and a carbonate wash play in the Sierra Madre Oriental of Mexico. They found that GTM provided not only the most likely cluster associated with a given voxel, but also the probability that that voxel belongs each of clusters, providing a measure of confidence or risk in the prediction.

*K*-means, SOM, and GTM are all unsupervised learning techniques, where the clustering is driven only by the choice of input attributes and the number of desired clusters. If we wish to teach the computer to mimic the facies identification previously chosen by a skilled interpreter, or link seismic facies to electro-facies interpreted using wireline logs, we need to introduce “supervision” or external control to the clustering algorithm. The most popular means of supervised learning classification are based on artificial neural networks (ANN). Meldahl et al. (1999) used seismic energy and coherence attributes coupled with interpreter control (picked seed points) to train a neural network to identify hydrocarbon chimneys. West et al. (2002) used a similar workflow where the objective was seismic facies analysis of a channel system and the input attributes were textures. Corradi et al. (2009) used GLCM (gray level co-occurrence matrix)
textures and ANN, with controls based on wells and skilled interpretation of some key 2D vertical slices to map sand, evaporate, and sealing vs. non-sealing shale facies offshore west Africa.

Support vector machine (SVM, where the word “machine” is due to Turing’s (1950) mechanical decryption machine) is a more recent introduction to (e.g. Li and Castagna, 2004; Kuzma and Rector, 2004, 2005; Zhao et al., 2005; Al-Anazi and Gates, 2010). Originating from maximum margin classifiers, SVMs have gained great popularity for solving pattern classification and regression problems since the concept of a “soft margin” was first introduced by Cortes and Vapnik (1995). SVMs map the $N$-dimensional input data into a higher dimensional latent (often called feature) space, where clusters can be linearly separated by hyperplanes. Detailed description on SVMs can be found in Cortes and Vapnik (1995), Cristianini and Shawe-Taylor (2000), and Schölkopf and Smola (2002). Li and Castagna (2004) used SVM to discriminate alternative AVO responses while Zhao et al. (2014) and Zhang et al. (2015) used a variation of SVM using mineralogy logs and seismic attributes to predict lithology and brittleness in a shale resource play.

We begin our paper by providing a summary of the more common clustering techniques used in seismic facies classification, emphasizing their similarities and differences. We start from the unsupervised learning $k$-means algorithm, progress through projections onto principal component hyperplanes, and end with projections onto SOM and GTM manifolds, which are topological spaces that resemble Euclidean space near each point. Next, we provide a summary of supervised learning techniques including artificial neural networks and support vector machines. Given these definitions, we apply each of these methods to identify seismic facies in the same data volume acquired in the Canterbury Basin of New Zealand. We conclude with a discussion on the advantages and limitations of each method and areas for future algorithm development and
workflow refinement. At the very end, we provide an appendix containing some of the mathematical details to better quantify how each algorithm works.

**REVIEW OF UNSUPERVISED LEARNING CLASSIFICATION TECHNIQUES**

*Crossplotting*

Crossplotting one or more attributes against each other is an interactive and perhaps the most common clustering technique. In its simplest implementation, one computes and then displays a 2D histogram of two attributes. In most software packages, the interpreter then identifies a cluster of interest and draws a polygon around it. While several software packages allow crossplotting of up to three attributes, crossplotting more than three attributes quickly becomes intractable. One workflow to address this visualization limitation is to first project a high number of attributes onto the first two or three eigenvectors, and then crossplot the principal components. Principal components will be discussed later in the section on projection methods.

*K-means classification*

K-means (MacQueen, 1967) is perhaps the simplest clustering algorithm and is widely available in commercial interpretation software packages. The method is summarized in the cartoons shown in Figure 2. One drawback of the method is that the interpreter needs to define how many clusters reside in the data. Once the number of clusters is defined, the cluster means or centers are defined either on a grid or randomly to begin the iteration loop. Since attributes have different units of measurement (e.g. Hz for peak frequency, 1/km for curvature, and mV for RMS amplitude) the distance of each data point to the current means are computed by scaling the data by the inverse of the covariance matrix, giving us the “Mahalanobis” distance (see Appendix).
Each data point is then assigned to the cluster to whose mean it is closest. Once assigned, new cluster means are computed from the newly assigned data clusters and the process repeated. If there are $Q$ clusters, the process will converge in about $Q$ iterations.

$K$-means is fast and easy to implement. Unfortunately, the clustering has no structure such that there is no relationship between the cluster numbering (and therefore coloring) and the proximity of one cluster to another. This lack of organization can result in similar facies appearing in totally different colors, confusing the interpretation. Tuning the number of clusters to force similar facies into the same cluster is a somewhat tedious procedure that also decreases the resolution of the facies map.

**Projection Techniques**

Although not defined this way in the pattern recognition literature, since this is a tutorial, we will lump the following methods, principal component analysis (PCA), self-organizing maps, and generative topographic maps together and call them “projection techniques”. Projection techniques project data residing in a higher dimensional space (say a 5D space defined by five attributes) onto a lower dimensional space (say a 2D plane or deformed 2D surface). Once projected, the data can be clustered in that space by the algorithm (such as SOM) or interactively clustered by the interpreter by drawing polygons (routine for PCA, and our preferred analysis technique for both SOM and GTM).

**Principal Component Analysis**

Principal component analysis is widely used to reduce the redundancy and excess dimensionality of the input attribute data. Such reduction is based on the assumption that most of
the signals are preserved in the first few principle components (eigenvectors), while the last principal components contain uncorrelated noise. In this tutorial, we will use PCA as the first iteration of the SOM and GTM algorithms. Many workers use PCA to reduce redundant attributes into “meta attributes” to simplify the computation. The first eigenvector is a vector in $N$-dimensional attribute space that best represents the attribute patterns in the data. Cross-correlating (projecting) the $N$-dimensional data against the first eigenvector at each voxel gives us the first principal component volume. If we scale the first eigenvector by the first principal component and subtract it from the original data vector, we obtain a residual data vector. The second eigenvector is that vector that best represents the attribute patterns in this residual. Cross-correlating (projecting) the second eigenvector against either the original data or residual data vector at each voxel gives us the second principal component volume. This process continues for all $N$-dimensions resulting in $N$ eigenvectors and $N$ principal components. In this paper, we will limit ourselves to the first two eigenvectors which thus define the plane that least-squares fits the $N$-dimensional attribute data. Figure 3c shows a numerical example of the first two principle components defining a plane in a 3-dimensional data space.

Self-organizing maps

While many workers (e.g. Coleou et al., 2003) describe SOM as a type of neural network, for the purposes of this tutorial, we prefer to describe SOM as a manifold projection technique. Kohonen (1982) SOM, originally developed for gene pattern recognition, is one of the most popular classification techniques, and it has been implemented in at least four commercial software packages for seismic facies classification. The major advantage of SOM over $k$-means is that the clusters residing on the deformed manifold in $N$-dimensional data space are directly mapped to a
rectilinear or otherwise regularly gridded latent space. We provide a brief summary of the mathematical formulations of the SOM implementation used in this study in the Appendix.

Although SOM is one of the most popular classification technique there are several limitations to the SOM algorithm. First, the choice of neighborhood function at each iteration is subjective, with different choices resulting in different solutions. Second, the absence of a quantitative error measure does not let us know whether the solution has converged to an acceptable level, thus providing confidence in the resulting analysis. Third, while we find the most likely cluster for a given data vector, we have no quantitative measure of confidence in the facies classification, and no indication if the vector could be nearly as well represented by other facies.

*Generative topographic mapping*

GTM is a nonlinear dimensionality reduction technique that provides a probabilistic representation of the data vectors on a lower \( L \)-dimensional deformed manifold that is in turn mapped to an \( L \)-dimensional latent space. While SOM seeks the node or prototype vector that is closest to the randomly chosen vector from the training or input dataset, in GTM each of the nodes lying on the lower dimensional manifold provides some mathematical support to the data and is considered to be to some degree “responsible” for the data vector (Figure 4). The level of support or “responsibility” is modeled with a constrained mixture of Gaussians. The model parameter estimations are determined by maximum likelihood using the Expectation Maximization (EM) algorithm (Bishop et al., 1998).

Because GTM theory is deeply rooted in probability, it can also be used in modern risk analysis. We can extend the GTM application in seismic exploration by projecting the mean posterior probabilities of a particular window of multiattribute data (say, about a producing well)
onto the 2D latent space. By project the data vector at any given voxel onto the latent space, we obtain a probability estimates of whether it falls into the same category (Roy et al., 2014). We thus have a probabilistic estimate of how similar any data vector is to attribute behavior (and hence facies) about a producing or non-producing well of interest.

**Other Unsupervised Learning Methods**

There are many other unsupervised learning techniques, several of which were evaluated by Barnes and Laughlin (2002). We do not currently have access to software to apply independent component analysis and Gaussian mixture models to our seismic facies classification problem, but mention them as possible candidates.

*Independent component analysis*

Like PCA, independent component analysis (ICA) is a statistical technique used to project a set of $N$-dimensional vectors onto a smaller $L$-dimensional space. Unlike PCA which is based on Gaussian statistics, whereby the first eigenvector best represents the variance in the multidimensional data, ICA attempts to project data onto subspaces that result in non-Gaussian distributions which are then easier to separate and visualize. Honorio et al. (2014) successfully apply ICA to multiple spectral components to delineate architectural elements of an offshore Brazil carbonate terrain. Both PCA and ICA are commonly used to reduce a redundant set of attributes to form a smaller set of independent meta-attributes (e.g. Gao, 2007).

*Gaussian mixture models*

Gaussian mixture model (GMM), are parametric models of probability distributions which can provide greater flexibility and precision in modeling than traditional unsupervised clustering
algorithms. Lubo et al. (2014) apply this technique to a suite of well logs acquired over Horseshoe Atoll, west Texas, to generate different lithologies. These GMM lithologies are then used to calibrate 3D seismic prestack inversion results to generate a 3D rock property model. At present, we do not know of any GMM algorithms applied to seismic facies classification using seismic attributes as input data.

**REVIEW OF SUPERVISED LEARNING CLASSIFICATION TECHNIQUES**

*Artificial Neural Networks*

Artificial neural networks can be used in both unsupervised and supervised mulitattribute analysis (van der Baan and Jutten, 2000). The multilayer perceptron (MLP) and the radial basis function (RBF) are two popular types of neural networks used in supervised learning. Probabilistic neural network, PNN, which also uses radial basis functions, forms the basis of additional neural network geophysical applications. In terms of network architecture, the supervised algorithms are feed-forward networks. In contrast, the unsupervised SOM algorithm described earlier is a recurrent (or feed-backward) network. An advantage of feed-forward networks over SOMs is the ability to predict both continuous values (such as porosity) as well as discrete values (such as facies class number). Applications of neural networks can be found in seismic inversion (Rôth and Tarantola, 1994), well log prediction from other logs (Huang et al., 1996; Lim, 2005), waveform recognition (Murat and Rudman, 1992), seismic facies analysis (West et al., 2002), and reservoir property prediction using seismic attributes (Yu et al., 2008; Zhao and Ramachandran, 2013). For the last application listed above, however, due to the resolution difference between seismic and well logs, structural and lithologic variation of inter-well points, and the highly nonlinear relation between these two domains, achieving a convincing prediction result can be challenging. In this
case, geostatistical methods such as Bayesian analysis can be used jointly to provide a probability index, giving interpreters an estimate of how much confidence they should have in the prediction.

Artificial neural networks are routinely used in the exploration and production industry. ANN provides a means to correlate well measurements such as gamma ray logs to seismic attributes (e.g. Verma, 2012) where the underlying relationship is a function of rock properties, depositional environment, and diagenetic alteration. Although it has produced reliable classification in many applications during its service, defects such as converging to local minima and difficult in parameterization are not negligible. In both industrial and scientific applications, we prefer a constant and robust classifier once the training vectors and model parameters have been determined. This leads to the more recent supervised learning technique developed in the late 20th century, the support vector machines.

**Support Vector Machines**

The basic idea of SVMs is straightforward. First, we transform the training data vectors into a still higher dimensional “feature” space using nonlinear mapping. Then we find a hyperplane in this feature space that separates the data into two classes with an optimal “margin”. The concept of a margin is defined to be the smallest distance between the separation hyperplane (commonly called a decision boundary) and the training vectors (Bishop, 2006) (Figure 5). An optimal margin balances two criteria: maximizing the margin, thereby giving the classifier the best generalization, and minimizing the number of misclassified training vectors if the training data are not linearly separable. The margin can also be described as the distance between the decision boundary and two hyperplanes defined by the data vectors which have the smallest distance to the decision boundary. These two hyperplanes are called the “plus-plane” and the “minus-plane”. The vectors
which lie exactly on these two hyperplanes mathematically define or “support” them and are called support vectors. Tong and Koller (2002) show that the decision boundary is dependent solely on the support vectors, resulting in the name “support vector machines”.

SVMs can be used in either a supervised or in a semi-supervised learning mode. In contrast to supervised learning, semi-supervised learning defines a learning process that utilizes both labeled and unlabeled vectors. When there are a limited number of interpreter classified data vectors, the classifier may not act well due to insufficient training. In semi-supervised training, some of the nearby unclassified data vectors are automatically selected and classified based on a distance measurement during the training step, as in an unsupervised learning process. These vectors are then used as additional training vectors (Figure 6), resulting in a classifier that will perform better for the specific problem. The generalization power is sacrificed by using unlabeled data. In this tutorial we focus on SVM; however, the future of semi-supervised SVM in geophysical applications is quite promising.

Proximal Support Vector Machines

Proximal support vector machine (PSVM) (Fung and Mangasarian, 2001, 2005) is a recent variant of SVM, which, instead of looking for a separating plane directly, builds two parallel planes that approximate two data classes; the decision-boundary then falls between these two planes (Figure 7). Other researchers have found that PSVM provides comparable classification correctness to standard SVM but at considerable computational savings (Fung and Mangasarian, 2001, 2005; Mangasarian and Wild, 2006). In this tutorial, we use PSVM as our implementation of SVM. Details on the PSVM algorithm are provided in the Appendix.
We may face problems in seismic interpretation that are linearly inseparable in the original input multidimensional attribute space. In SVM, we map the data vectors into a higher dimensional space where they become linearly separable (Figure 8), where the increase in dimensionality may result in significantly increased computational cost. Instead of using an explicit mapping function to map input data into a higher dimensional space, PSVM achieves the same goal by manipulating a kernel function in the input attribute space. In our implementation, we use a Gaussian kernel function, but in principal many other functions can be used (Shawe-Taylor and Cristianini, 2004).

SVM can be used either as a classifier or as a regression operator. Used as a regression operator, SVM is capable of predicting petrophysical properties such as porosity (Wong et al., 2005), $V_p$, $V_s$ and density (Kuzma and Rector, 2004), and permeability (Al-Anazi and Gates, 2010; Nazari et al., 2011). In all such applications, SVM shows comparable or superior performance to neural networks with respect to prediction error and training cost. When used as a classifier, SVM is suitable in predicting lithofacies (Al-Anazi and Gates, 2010; Torres and Reveron, 2013; Wang et al., 2014; Zhao et al., 2014) or pseudo rock properties (Zhang et al., 2015), either from well log data, core data, or seismic attributes.

GEOLGIC SETTING

In this tutorial we utilized the Waka-3D seismic survey acquired over the Canterbury Basin, offshore New Zealand, generously made public by New Zealand Petroleum and Minerals. Readers can request this data set through their website for research purposes. Figure 9 shows the location of this survey, where the red rectangle corresponds to time slices shown in subsequent figures. The study area lies on the transition zone of continental slope and rise, with abundance of paleocanyons and turbidite deposits of Cretaceous and Tertiary ages. These sediments are
deposited in a single, tectonically driven transgressive – regressive cycle (Uruski, 2010). Being a very recent and underexplored prospect, publically available comprehensive studies of the Canterbury Basin are somewhat limited. The modern seafloor canyons shown in Figure 9 are good analogs of the deeper paleocanyons illuminated by the 3D seismic amplitude and attribute data.

**ATTRIBUTE SELECTION**

In their comparison of alternative unsupervised learning techniques, Barnes and Laughlin (2002) concluded that the appropriate choice of attributes was the most critical component of computer assisted seismic facies identification. Although interpreters are skilled at identifying facies, such recognition is often subconscious and hard to define (see Eagleman’s 2012 discussion on differentiating male from female chicks and identifying military aircraft from silhouettes). In supervised learning, the software does some of the work during the training process, though we must always be wary of false correlations if we provide too many attributes (Kalkomey, 1999). For the prediction of continuous data such as porosity, Russell (1997) and others suggest that one begin with exploratory data analysis, where one simply cross-correlates a candidate attribute with the desired property at the well. Such cross-correlation does not work well when trying to identify seismic facies, which are simply “labeled” with an integer number or alphanumeric name.

Table 1 summarizes how we four interpreters perceive each of the seismic facies of interest. Once we have enumerated the seismic expression, the quantification using attribute expression is relatively straightforward. In general, amplitude and frequency attributes are lithology indicators and may provide direct hydrocarbon detection in conventional reservoirs, geometric attributes delineate reflector morphology such as dip, curvature, rotation, and convergence, while statistical and texture attributes provides information about data distribution that quantifies subtle patterns.
that are hard to define (Chopra and Marfurt, 2007). Attributes such as coherence provide images of the edges of seismic facies rather than a measure of the facies themselves, although slumps often appear as a suite of closely spaced faults separating rotated fault blocks. Finally, what we see as interpreters and what our clustering algorithms see can be quite different. While we may see a slump feature as exhibiting a high number of faults per km, our clustering algorithms are applied voxel by voxel and see only the local behavior. Extending the clustering to see such large scale textures requires the development of new texture attributes.

The number of attributes should be as small as possible to discriminate the facies of interest, and each attribute should be mathematical independent from the others. While it may be fairly easy to represent three attributes with a deformed 2D manifold, increasing the dimensionality results in increased deformation, such that our manifold may fold on itself or may not accurately represent the increased data variability. Because the Waka-3D survey is new to all four authors, we have tested numerous attributes that we think may highlight different facies in the turbidite system. Among these attributes, we find the shape index to be good for visual classification but dominates the unsupervised classifications with valley and ridge features across the survey. After such analysis we chose four attributes that are mathematically independent but should be coupled through the underlying geology: peak spectral frequency, peak spectral magnitude, GLCM homogeneity, and curvedness, as the input to our classifiers. The peak spectral frequency and peak spectral magnitude form an attribute pair that crudely represents the spectral response. Peak frequency of spectrally whitened data is sensitive to tuning thickness while peak magnitude is a function of both tuning thickness and impedance contrast. GLCM homogeneity is a texture attribute that has a high value for adjacent traces with similar (high or low) amplitudes and measures the continuity of a seismic facies. Curvedness defines the magnitude of reflector
structural or stratigraphic deformation, with dome-, ridge-, saddle-, valley-, and bowl-shaped features exhibiting high curvedness and planar features exhibiting zero curvedness.

Figure 10 shows a time slice at $t=1.88$ s through the seismic amplitude volume on which we identify channels (white arrows), high amplitude deposits (yellow arrows), and slope fans (red arrows). Figure 11 shows an equivalent time slice through peak spectral frequency co-rendered with peak spectral magnitude that emphasizes the relative thickness and reflectivity of the turbidite system and surrounding slope fan sediments into which it was incised. The edges of the channels are delineated by Sobel filter similarity. We show equivalent time slices through (Figure 12) GLCM homogeneity, and (Figure 13) co-rendered shape index and curvedness. In Figure 14 we show a representative vertical slice at line AA’ in Figure 10 cutting through the channels through (Figure 14a) seismic amplitude, (Figure 14b) seismic amplitude co-rendered with peak spectral magnitude/peak spectral frequency, (Figure 14c) seismic amplitude co-rendered with GLCM homogeneity, and (Figure 14d) seismic amplitude co-rendered shape index and curvedness. White arrows indicate incised valleys, yellow arrows high amplitude deposits, and red arrows a slope fan. We note several of the incised values are visible at time slice $t=1.88$ s.

In a conventional interpretation workflow, the geoscientist would examine each of these attribute images and integrate them within a depositional framework. Such interpretation takes time and may be impractical for extremely large data volumes. In contrast, in seismic facies classification the computer either attempts to classify what it sees as distinct seismic facies (in unsupervised learning) or attempts to emulate the interpreter’s classification made on a finite number of vertical sections, time, and/or horizon slices and apply the same classification to the full 3D volume (in supervised learning). In both cases, the interpreter needs to validate the final classification to determine if they represent seismic facies of interest. In our example we will use
Sobel filter similarity to separate the facies and then evaluate how they fit within our understanding of a turbidite system.

APPLICATION

Given these four attributes, we now construct four-dimensional attribute vectors as input to the previously described classification algorithms. To better illustrate the performance of each algorithm, we summarize the data size, number of computational processors, and runtime in Table 2. All the algorithms are developed by the authors except ANN, which is implemented using MATLAB® toolbox.

We begin with $k$-means. As previously discussed, a limitation of $k$-means is the lack of any structure to the cluster number selection process. We illustrate this limitation by computing $k$-means with 16 (Figure 15) and 256 (Figure 16) clusters. On Figure 15, we can identify high amplitude overbank deposits (yellow arrows), channels (white arrows), and slope fan deposits (red arrows). A main limitation of $k$-means is that there is no structure linking the clusters, which leads to a somewhat random choice of color assignment to clusters. This problem becomes more serious when more clusters are selected: the result with 256 clusters (Figure 16) is so chaotic that we can rarely separate the overbank high amplitude deposits (yellow arrows) and slope fan deposits (red arrows) that were easily separable in Figure 15. For this reason, modern $k$-means applications focus on estimating the correct number of clusters in the data.

In contrast to $k$-means, SOM restricts the cluster centers to lie on a deformed 2D manifold. While clusters may move closer or further apart, they still form (in our implementation) a deformed quadrilateral mesh which maps to a rectangular mesh on the 2D latent space. Mapping the latent space to a continuous 1D (Coleou et al., 2003) or 2D color bar (Strecker and Uden, 2002), reduces
the sensitivity to the number of clusters chosen. We follow Gao (2007) and avoid guessing at the number of clusters necessary to represent the data by overdefining the number of prototype vectors to be 256 (the limit of color levels in our commercial display software). These 256 prototype vectors (potential clusters) reduce to only three or four distinct “natural” clusters through the SOM neighborhood training criteria. The 2D SOM manifold is initialized using the first two principle components, defining a plane through the \( N \)-dimensional attribute space (Figure 17). The algorithm then deforms the manifold to better fit the data. Overdefining the number of prototype vectors results in clumping into a smaller number natural clusters. These clumped prototype vectors project onto adjacent locations in the latent space are therefore appear as subtle shades of the same color as indicated by the limited palette of 256 colors shown in Figure 18. On the classification result shown on Figure 18, we can clearly identify the green colored spill-over deposits (yellow arrows). The difference between channel fill (white arrows) and slope fans (red arrows) is insignificant. However, by co-rendering with similarity, the channels are delineated nicely, allowing us to visually distinguish channel fills and the surrounded slope fans. We can also identify some purple color clusters (orange arrows) which we interpret to be crevasse splays.

Next, we apply GTM to the same four attributes. We compute two “orthogonal” projections of data onto the manifold and thence onto the two dimensions of the latent space (Figure A2). Rather than define explicit clusters, we project the mean a posteriori probability distribution onto the 2D latent space and then export the projection onto the two latent space axes. We crossplot the projections along axes 1 and 2 and map them against a 2D color bar (Figure 19). In this slice, we see channels delineated by purple colors (white arrows), point bar and crevasse splays in pinkish colors (yellow arrows), and slope fans in lime green colors (red arrows). We can also identify some thin, braided channels at the south end of the survey (blue arrow). Similarly to the SOM result,
similarity separates the incised valleys from the slope fans. However, the geological meaning of the orange colored facies is somehow vague. This is the nature of unsupervised learning techniques in that the clusters represent topological differences in the input data vectors, which are not necessarily the facies differences we wish to delineate. We can ameliorate this shortcoming by adding a posteriori supervision to the GTM manifold. The simplest way to add supervision is to compute the average attribute vectors about a given seismic facies and map it to the GTM crossplot. Then, the interpreter can manually define clusters on the 2D histogram by constructing one or more polygons (Figure 20), where we cluster the data into four facies: multistoried channels (blue), high-energy point bar and crevasse splay deposits (yellow), slope fans (green), and “everything else” (red). A more quantitative methodology is to mathematically project these average clusters onto the manifold, and then cross multiply the probability distribution of the control vectors against the probability distribution function of each data vector, thereby forming the Bhattacharya distance (Roy et al., 2013, 2014). Such measures then provide a probability ranging between 0 and 100% as to whether the data vector at any seismic sample point is like the data vectors about well control (Roy et al., 2013, 2014) or like the average data vector within a facies picked by the interpreter.

The a posteriori supervision added to GTM is the critical prior supervision necessary for supervised classification such as ANN (Figure A3) and SVM (Figure A4). In this study we used the same four attributes as input for both unsupervised and supervised learning techniques. Our supervision consists of picked seed points for the three main facies previously delineated using the unsupervised classification results, which are multistoried channel, point bar and crevasse splay deposits, and slope fans, plus an additional channel flank facies. The seed points are shown in Figure 21. Seed points should be picked with great caution to correctly represent the corresponding
facies, any false picking (a seed point that does not belong to the intended facies) will greatly compromise the classification result. We then compute averages of the four input attributes within a 7 trace X 7 trace X 24 ms window about each seed point to generate a training table which consists of 4-dimensional input attribute vectors and one dimensional targets (the labeled facies).

For our ANN application, we used the neural networks toolbox in MATLAB®, and generated a probabilistic neural network (PNN) composed of 20 neurons. Because of the relatively small size of the training data, the training process only took a second or so; however, since a PNN may converge to local minima, we are not confident that our first trained network has the best performance. Our workflow is then to rerun the training process 50 times and choose the network exhibiting the lowest training and cross-validation errors. Figures 22 and 23 show the PNN performance during training, while Figure 24 shows the PNN classification result. We notice that all the training, testing, and cross-validation performance are acceptable, with training and cross-validation correctness being around 90%, and testing correctness being over 86%. We identify blue channel stories within the relatively larger scale incised valleys (white arrows), and yellow point bars and crevasse splays (yellow arrows). However, many of the slope fan deposits are now classified as channel flanks or multistoried channels (blue arrows), which need to be further calibrated with well log data. Nevertheless, as a supervised learning technique, ANN provides classification with explicit geological meaning, which is its primary advantage over unsupervised learning techniques.

Finally, we cluster our four-dimensional input data using SVM, using the same training data (interpreter picks) as for ANN. The workflow is similar to ANN in that we ran 20 passes of training, varying the Gaussian kernel standard deviation, $\sigma$, and misclassification tolerance, $\varepsilon$, parameters for each pass. These parameter choices are easier than selecting the number of neurons
for ANN, since the SVM algorithm solves a convex optimization problem that converges to a global minima. The training and cross-validation performance is comparable to ANN, with roughly 92% training correctness and 85% cross-validation correctness. Figure 25 shows the SVM classification result at time $t = 1.88 \text{s}$. The SVM map follows the same pattern as we have seen on the ANN map, but is generally cleaner, with some differences in details. Compared to ANN, SVM successfully mapped more of the slope fans (white arrows), but missed some crevasse splays that were correctly picked by ANN (yellow arrow). We also see a great amount of facies variation within the incised valleys, which is reasonable because of the multiple course changes of a paleochannel during its deposition that results in multiple channel stories. Finally, we note some red lines following NW-SE direction (red arrows) which correspond to acquisition footprint.

**CONCLUSIONS**

In this paper we have compared and contrasted some of the more important multiattribute facies classification tools, including four unsupervised (PCA, $k$-means, SOM, GTM) and two supervised (ANN, SVM) learning techniques. In addition to highlighting the differences in assumptions and implementation, we have applied each method to the same Canterbury Basin survey, with the goal of delineating seismic facies in a turbidite system to demonstrate the effectiveness and weaknesses of each method. $K$-means and SOM move the user-defined number of cluster centers towards the input data vectors. PCA is the simplest manifold method, where the data variability in our examples is approximated by a 2D plane defined by the first two eigenvectors. GTM is more accurately described as a mapping technique, like PCA, where the clusters are formed either in the human brain as part of visualization or through crossplotting and the construction of polygons. SOM and GTM manifolds deform to fit the N-dimensional data. In SOM, the cluster centers (prototype vectors) move along the manifold towards the data vectors,
forming true clusters. In all four methods, any labeling of a given cluster to a given facies happens after the process is completed. In contrast, ANN and SVM build a specific relation between the input data vectors and a subset of user-labeled input training data vectors thereby explicitly labeling the output clusters to the desired facies. Supervised learning is constructed from a limited group of training samples (usually at certain well locations or manually picked seed points) which generally are insufficient to represent all the lithologic and stratigraphic variations within a relatively large seismic data volume. A pitfall of supervised learning is that unforeseen clusters will be misclassified as clusters that have been chosen.

For this reason, unsupervised classification products can be used to construct not only an initial estimate of the number of classes, but also a validation tool to determine if separate clusters have been incorrectly lumped together. We advise computing unsupervised SOM or GTM prior to picking seed points for subsequent supervised learning, to clarify the topological differences mapped by our choice of attributes. Such mapping will greatly improve the picking confidence, because the seed points are now confirmed by both human experience and mathematical statistics.

The choice of the correct suite of attributes is critical. Specifically, images that are ideal for multiattribute visualization may be suboptimal for clustering. We made several poor choices in previous iterations of writing this paper. The image of inline (SW-NE) structural dip illustrates this problem directly. While a skilled interpreter sees a great deal of detail in Figure 26, there is no clear facies difference between positive and negative dips, such that this component of vector dip cannot be used to differentiate them. A better choice would be dip magnitude, except that a long wavelength overprint (such as descending into the basin) would again bias our clustering in a manner that is unrelated to facies. Therefore, we tried to use relative changes in dip – curvedness
and shape indices measure lateral changes in dip, and reflector convergence which differentiates conformal from nonconformal reflectors.

Certain attributes should never be used in clustering. Phase, azimuth, and strike have circular distributions, where a phase value of -180 indicates the same value as +180. No trend can be found. While the shape index, \( s \), is not circular, ranging between -1 and +1, the histogram has a peaks about the ridge (\( s=+0.5 \)) and about the valley (\( s=-0.5 \)). We speculate that shape components may be more amenable to classification. Reflector convergence follows the same pattern as curvedness. For this reason we only used curvedness as a representative of these three attributes. The addition of this choice improved our clustering.

Edge attributes like the Sobel filter similarity and coherence are not useful for the example show here; instead, we have visually added them as an edge “cluster” and co-rendered with the images shown in Figures 15-21, 24, and 25. In contrast, when analyzing more chaotic features such as salt domes and karst collapse, coherence is a good input to clustering algorithms. We do wish to provide an estimate of continuity and randomness to our clustering. To do so, we follow Corradi et al. (2009) and West et al. (2002) and use GLCM homogeneity as an input attribute.

Theoretically, no one technique is superior to all the others in every aspect, and each technique has its inherent advantages and defects. \( K \)-means with a relatively small numbers of clusters is the easiest algorithm to implement, provides rapid interpretation, but lacks the relation among clusters. SOM provides a generally more “interpreter friendly” clustering result with topological connections among clusters, but is computationally more demanding than \( k \)-means. GTM relies on probability theory and enables the interpreter to add posteriori supervision by manipulating the data’s posterior probability distribution; however, it is not widely accessible to
the exploration geophysicist community. Rather than displaying the conventional cluster numbers (or labels), we suggest displaying the cluster coordinates projected onto the 2D SOM and GTM latent space axes. Doing so not only provides greater flexibility in constructing a 2D color bar but also provides data that can be further manipulated using 2D crossplot tools.

For the two supervised learning techniques, ANN suffers from the convergence problem and requires expertise to achieve the optimal performance, while the computation cost is relatively low. SVM is mathematically more robust and easier to train, but is more computationally demanding.

Practically, if no software limitations are set, we can make suggestions on how an interpreter can incorporate these techniques to facilitate seismic facies interpretation at different exploration and development stages. To identify the main features in a recently acquired 3D seismic survey on which limited to no traditional structural interpretation is done, $k$-means is a good candidate for exploratory classification starting with a small $K$ (typically $K = 4$) and gradually increase the number of class. As more data are acquired (e.g. well log data and production data) and detailed structural interpretation has been performed, SOM or GTM focusing in the target formations will provide more refined classification, which needs to be calibrated with wells. In the development stage when most of the data have been acquired, with proper training process, ANN and SVM provide targeted products, characterizing the reservoir by mimicking interpreters’ behavior. Generally, SVM provides superior classification than ANN but at a considerably higher computational cost, so choosing between these two requires balancing performance and runtime cost. As a practical manner, no given interpretation software platform provides all five of these clustering techniques, such that many of the choices are based on software availability.
Because we wish this paper to serve as an inspiration of interpreters, we do want to reveal one drawback of our work: all the classifications are performed volumetrically but not along a certain formation. Such classification may be biased by the bonding formations above and below the target formation (if we do have a target formation), therefore contaminates the facies map. However, we want to make the point that such classification can happen at a very early stage of interpretation, when both structural interpretation and well logs are very limited. And even in such situation, we can still use classification techniques to generate facies volumes to assist subsequent interpretation.

In the 1970s and 1980s much of geophysical innovation in seismic processing and interpretation was facilitated by the rapid evolution of computer technology – from mainframes to minicomputers to workstations to distributed processing. We believe similar advances in facies analysis will be facilitated by the rapid innovation in “big data” analysis, driven by needs in marketing and security. While we may not answer Turing’s (1950) question “Can machines think?”, we will certainly be able to teach them how to emulate a skilled human interpreter.

ACKNOWLEDGEMENTS

We thank New Zealand Petroleum and Minerals for providing the Waka-3D seismic data to the public. Financial support for this effort and for the development of our own $k$-means, PCA, SOM, GTM, and SVM algorithms was provided by the industry sponsors of the Attribute-Assisted Seismic Processing and Interpretation (AASPI) consortium in the University of Oklahoma. The ANN examples were run using MATLAB®. We thank colleagues Fangyu Li, Sumit Verma, Lennon Infante, and Brad Wallet for their valuable input and suggestions. All the 3D seismic displays were made using licenses to Petrel, provided to the University of Oklahoma for research
and education courtesy of Schlumberger. Finally, we want to express our great respect to the people that have contributed the development of pattern recognition techniques in exploration geophysics field.
APPENDIX: Mathematical details

In this appendix we summarize many of the mathematical details defining the various algorithm implementations. Although insufficient to allow a straightforward implementation of each algorithm, we hope to more quantitatively illustrate the algorithmic assumptions as well as algorithmic similarities and differences. Because k-means and artificial neural networks have been widely studied, in this appendix we only give some principle statistical background, and brief reviews of SOM, GTM, and PSVM algorithms involved in this tutorial. We begin this appendix by giving statistical formulations of the covariance matrix, principal components and the Mahalanobis distance when applied to seismic attributes. We further illustrate the formulations and some necessary theory for SOM, GTM, ANN, and PSVM. Because of the extensive use of mathematical symbols and notations, a table of shared mathematical notations is given in Table A1. All other symbols are defined in the text.

Covariance matrix, principal components, and the Mahalanobis distance

Given a suite of $N$ attributes, the covariance matrix is defined as

$$ C_{mn} = \frac{1}{J} \sum_{j=1}^{J} (a_{jm}(t_j,x_j,y_j) - \mu_m)(a_{jn}(t_j,x_j,y_j) - \mu_n), \quad (A - 1) $$

where $a_{jm}$ and $a_{jn}$ are the $m^{th}$ and $n^{th}$ attributes, $J$ is the total number of data vectors, and where

$$ \mu_n = \frac{1}{J} \sum_{j=1}^{J} a_{jn}(t_j,x_j,y_j), \quad (A - 2) $$

is the mean of the $n^{th}$ attribute. If we compute the eigenvalues, $\lambda_i$, and eigenvectors, $v_i$, of the real, symmetric covariance matrix, $C$, the $i^{th}$ principal component at data vector $j$ is defined as

$$ p_{ji} = \sum_{n=1}^{N} a_{jn}(t_j,x_j,y_j) v_{ni}, \quad (A - 3) $$
where \( v_{ni} \) indicates the \( n^{th} \) attribute component of the \( i^{th} \) eigenvector. In this paper, the first two eigenvectors and eigenvalues are also used to construct an initial model in both the self-organizing maps (SOM) and generative topological mapping (GTM) algorithms.

The Mahalanobis distance, \( r_{jq} \), of the \( j^{th} \) sample from the \( q^{th} \) cluster center, \( \theta_q \), is defined as

\[
r_{jq}^2 = \sum_{n=1}^{N} \sum_{m=1}^{N} (a_{jn} - \theta_{nq})C_{nm}^{-1}(a_{jm} - \theta_{mq}),
\]

where the inversion of the covariance matrix, \( C \), takes place prior to extracting the \( mn^{th} \) element.

**Self-organizing maps**

Rather than computing the Mahalanobis distance, both SOM and GTM first normalize the data using a Z-scale. If the data exhibit an approximately Gaussian distribution, the Z-scale of the \( n^{th} \) attribute is obtained by subtracting the mean and dividing by the standard deviation (the square root of the diagonal of the covariance matrix, \( C_{nn} \)). To Z-scale non-Gaussian distributed data, such as coherence, one needs to first break the data using histograms that approximate a Gaussian. The objective of the SOM algorithm is to map the input seismic attributes onto a geometric manifold called the self-organized map. The SOM manifold is defined by a suite of prototype vectors \( \mathbf{m}_k \) lying on a lower-dimensional (in our case, 2D) surface which fit the \( N \)-dimensional attribute data. The prototype vectors \( \mathbf{m}_k \) are typically arranged in 2D hexagonal or rectangular structure maps that preserve their original neighborhood relationship, such that neighboring prototype vectors represent similar data vectors. The number of prototype vectors in the 2D map determines the effectiveness and generalization of the algorithm. One strategy is to estimate the number of initial clusters, and then to either divide or join clusters based on distance criteria. In our case, we follow Gao (2007) and overdefine the number of clusters to be the maximum number of colors supported.
by our visualization software. Interpreters then either use their color perception or construct polygons on 2D histograms to define a smaller number of clusters.

Our implementation of the SOM algorithm is summarized in Figure A1. After computing Z-scores of the input data, the initial manifold is defined to be a plane defined by the two first principal components. Prototype vectors \( \mathbf{m}_k \) are defined on a rectangular grid to the first two eigenvalues to range between \( \pm 2(\lambda_1)^{1/2} \) and \( \pm 2(\lambda_2)^{1/2} \). The seismic attribute data are then compared to each of the prototype vectors, finding the nearest one. This prototype vector and its nearest neighbors (those that fall within a range \( \sigma \), defining a Gaussian perturbation) are moved towards the data point. After all the training vectors have been examined, the neighborhood radius, \( \sigma \), is reduced. Iterations continue until \( \sigma \) approaches the distance between the original prototype vectors.

Given this background, Kohonen (2001) defines the SOM training algorithm using the following five steps:

\textit{Step 1:} Randomly chose a previously Z-scored input attribute vector, \( \mathbf{a}_j \), from the set of input vectors.

\textit{Step 2:} Compute the Euclidean distance between this vector \( \mathbf{a}_j \) and all prototype vectors \( \mathbf{m}_k \), \( k=1,2,...,K \). The prototype vector which has the minimum distance to the input vector \( \mathbf{a}_j \), is defined to be the “winner” or the Best Matching Unit, \( \mathbf{m}_b \):

\[
\big| \big| \mathbf{a}_j - \mathbf{m}_b \big| \big| = \min_k \big\{ \big| \big| \mathbf{a}_j - \mathbf{m}_k \big| \big| \big\}.
\] (A - 5)

\textit{Step 3:} Update the “winner” prototype vector and its neighbors. The updating rule for the weight of the \( k^{th} \) prototype vector inside and outside the neighborhood radius \( \sigma(t) \) is given by

\[
\mathbf{m}_k(t + 1) = \begin{cases} 
\mathbf{m}_k(t) + \alpha(t)h_{bk}(t)[\mathbf{a}_j - \mathbf{m}_k(t)], & \text{if } \big| \big| \mathbf{r}_k - \mathbf{r}_b \big| \big| \leq \sigma(t) \\
\mathbf{m}_k(t), & \text{if } \big| \big| \mathbf{r}_k - \mathbf{r}_b \big| \big| > \sigma(t)
\end{cases}
\] (A - 6)
where the neighborhood radius defined as $\sigma(t)$ is predefined for a problem and decreases with each iteration $t$. $\mathbf{r}_b$ and $\mathbf{r}_k$ are the position vectors of the winner prototype vector $\mathbf{m}_b$ and the $k^{th}$ prototype vector $\mathbf{m}_k$ respectively. We also define the neighborhood function, $h_{bk}(t)$, the exponential learning function, $\alpha(t)$, and the length of training, $T$. $h_{bk}(t)$ and $\alpha(t)$ decrease with each iteration in the learning process and are defined as

$$h_{bk}(t) = e^{-||\mathbf{r}_b - \mathbf{r}_k||^2/2\sigma^2(t)}, \quad (A - 7)$$

and

$$\alpha(t) = \alpha_0 \left(\frac{0.005}{\alpha_0}\right)^{t/T}. \quad (A - 8)$$

**Step 4:** Iterate through each learning step (steps 1-3) until the convergence criterion (which depends on the predefined lowest neighborhood radius and the minimum distance between the prototype vectors in the latent space) is reached.

**Step 5:** Project the prototype vectors onto the first two principal components and color code using a 2D color bar (Matos et al. 2009).

**Generative topological mapping**

In GTM, the grid points of our 2D deformed manifold in $N$-dimensional attribute space define the centers, $\mathbf{m}_k$, of Gaussian distributions of variance $\sigma^2 = \beta^{-1}$. These centers, $\mathbf{m}_k$, are in turn projected onto a 2D latent space, defined by a grid of nodes $\mathbf{u}_k$ and nonlinear basis functions, $\Phi$:

$$\mathbf{m}_k = \sum_{m=1}^{M} \mathbf{W}_{km} \Phi_m(\mathbf{u}_k), \quad (A - 11)$$

where $\mathbf{W}$ is a $K \times M$ matrix of unknown weights, $\Phi_m(\mathbf{u}_k)$ is a set of $M$ nonlinear basis functions, $\mathbf{m}_k$ are vectors defining the deformed manifold in the $N$-dimensional data space, and $k=1,2,...,K$ is the number of grid points arranged on a lower $L$-dimensional latent space (in our case, $L=2$). A noise model (the probability of the existence of a particular data vector $\mathbf{a}_j$ given weights $\mathbf{W}$ and
inverse variance $\beta$) is introduced for each measured data vector. The probability density function, $p$, is represented by a suite of $K$ radially symmetric $N$-dimensional Gaussian functions centered about $\mathbf{m}_k$ with variance of $1/\beta$:

$$p(\mathbf{a}_j | \mathbf{W}, \beta) = \sum_{k=1}^{K} \frac{1}{K} \left( \frac{\beta}{2\pi} \right)^{\frac{N}{2}} e^{-\frac{\beta}{2} |\mathbf{m}_k - \mathbf{a}_j|^2}. \quad (A - 12)$$

The prior probabilities of each of these components are assumed to be equal with a value of $1/K$, for all data vectors $\mathbf{a}_j$. Figure 4 illustrates the GTM mapping from an $L=2D$ latent space to the 3D data space.

The probability density model (GTM model) is fit to the data $\mathbf{a}_j$ to find the parameters $\mathbf{W}$ and $\beta$ using a maximum likelihood estimation. One of the popular techniques used in parameter estimations is the Expectation Maximization (EM) algorithm. Using Bayes’ theorem, and the current values of the GTM model parameters $\mathbf{W}$ and $\beta$, we calculate the $J \times K$ posterior probability or responsibility, $R_{jk}$, for each of the $K$ components in latent space for each data-vector:

$$R_{jk} = \frac{e^{-\frac{\beta}{2} |\mathbf{m}_k - \mathbf{a}_j|^2}}{\sum_i e^{-\frac{\beta}{2} |\mathbf{m}_i - \mathbf{a}_j|^2}}. \quad (A - 13)$$

Equation A-13 forms the “E-step” or Expectation step in the EM algorithm. The E-step is followed by the Maximization or “M-step”, which uses these responsibilities to update the model for a new weight matrix $\mathbf{W}$ by solving a set of linear equations (Dempster et al., 1977):

$$\left( \mathbf{\Phi}^T \mathbf{G} \mathbf{\Phi} + \frac{\alpha}{\beta} \mathbf{I} \right) \mathbf{W}^{new} = \mathbf{\Phi}^T \mathbf{R} \mathbf{X}, \quad (A - 14)$$

where

$$G_{kk} = \sum_{j=1}^{J} R_{jk}$$ are the non-zero elements of the $K \times K$ diagonal matrix $G$,

$\mathbf{\Phi}$ is a $K \times M$ Matrix with elements $\mathbf{\Phi} = \Phi_m(\mathbf{u}_k)$,
\( \alpha \) is a regularization constant to avoid division by zero, and 

\( \mathbf{I} \) is the \( M \times M \) identity matrix.

The updated value of \( \beta \) is given by

\[
\frac{1}{\beta_{\text{new}}} = \frac{1}{\mathcal{J} N} \sum_{j=1}^{\mathcal{J}} \sum_{k=1}^{K} R_{jk} \left\| \mathbf{W}_{km_{\text{new}}} \Phi_m(\mathbf{u}_k) - \mathbf{a}_j \right\|^2. \tag{A - 15}
\]

The initialization of \( \mathbf{W} \) is done so that the initial GTM model approximates the principal components (largest eigenvectors) of the input data, \( \mathbf{a}_j \). The value of \( \beta^{-1} \) is initialized to be the larger of the \((L+1)\)th eigenvalue from PCA where \( L \) is the dimension of the latent space. In Figure 4, \( L=2 \), such that we initialize \( \beta^{-1} \) to be the inverse of the third eigenvalue. Figure A2 summarizes this workflow.

**Artificial Neural Networks**

The artificial neural networks are a class of pattern recognition algorithms that were derived separately in different fields such as statistics and artificial intelligence. Artificial neural networks are easily accessible for most of the geophysical interpreters, so we only provide a general workflow of applying an ANN to seismic facies classification for completeness of this tutorial. The workflow is shown in Figure A3.

**Proximal support vector machines**

Because SVMs are originally developed to solve binary classification problems, the arithmetic we begin with a summary of the arithmetic describing a binary PSVM classifier.

Similarly to SVM, a PSVM decision condition is defined as (Figure 7):

\[
x^T \omega - \gamma \begin{cases} > 0, & x \in X^+; \\ = 0, & x \in X + \text{ or } X^-; \\ < 0, & x \in X^-, \end{cases} \tag{A - 16}
\]
where \( \mathbf{x} \) is an \( N \)-dimensional attribute vector to be classified, \( \mathbf{\omega} \) is a \( N \times I \) vector implicitly defines the normal of the decision-boundary in the higher dimensional space, \( \gamma \) defines the location of the decision-boundary, and “\( X^+ \)” and “\( X^- \)” indicate the two classes of the binary classification. PSVM solves an optimization problem and takes the form of (Fung and Mangasarian, 2001):

\[
\min_{\mathbf{\omega}, \gamma, y} \frac{1}{2} \|y\|^2 + \frac{1}{2} (\mathbf{\omega}^T \mathbf{\omega} + \gamma^2),
\]

subject to

\[
D(\mathbf{a} \mathbf{\omega} - e \gamma) + y = e.
\]

In this optimization problem, \( y \) is a \( J \times I \) error variable; \( \mathbf{a} \) is a \( J \times N \) sample matrix composed of \( J \) attribute vectors, which can be divided into two classes, “\( X^+ \)” and “\( X^- \)”. \( D \) is a \( J \times J \) diagonal matrix of labels with a diagonal composed of \( +1 \) for “\( X^+ \)” and \( -1 \) for “\( X^- \)”. \( \varepsilon \) is a non-negative parameter. Finally, \( e \) is a \( J \times I \) column vector of ones. This optimization problem can be solved by using a \( J \times I \) Lagrangian multiplier \( t \):

\[
L(\mathbf{\omega}, \gamma, y, t) = \varepsilon \frac{1}{2} \|y\|^2 + \frac{1}{2} (\mathbf{\omega}^T \mathbf{\omega} + \gamma^2) - t^T (D(\mathbf{a} \mathbf{\omega} - e \gamma) + y - e).
\]

By setting the gradients of \( L \) to zero, we obtain expressions for \( \mathbf{\omega}, \gamma \) and \( y \) explicitly in the knowns and \( t \), where \( t \) can further be represented by \( \mathbf{a}, D \) and \( \varepsilon \). Then by substituting \( \mathbf{\omega} \) in Equations A-17 and A-18 using its dual equivalent \( \mathbf{\omega} = \mathbf{a}^T D t \), we can arrive at (Fung and Mangasarian, 2001):

\[
\min_{\omega, \gamma, y} \frac{1}{2} \|y\|^2 + \frac{1}{2} (t^T t + \gamma^2),
\]

subject to

\[
D(aa^T D t - e \gamma) + y = e.
\]
Equations A-20 and A-21 provide a more desirable version of the optimization problem since one can now insert kernel methods to solve nonlinear classification problems made possible by the term $\mathbf{a}\mathbf{a}^T$ in Equation A-21. Utilizing the Lagrangian multiplier again (this time we denote the multiplier as $\mathbf{t}$), we can minimize the new optimization problem against $\mathbf{t}$, $\gamma$, $\mathbf{y}$ and $\mathbf{t}$. By setting the gradients of these four variables to zero, we can express $\mathbf{t}$, $\gamma$ and $\mathbf{y}$ explicitly by $\mathbf{t}$ and other knowns, where $\mathbf{t}$ is solely a dependent on the data matrices. Then for $N$-dimensional attribute vector $\mathbf{x}$ we write the decision conditions as

$$
\mathbf{x}^\top \mathbf{a}^\top \mathbf{D} \mathbf{t} - \gamma \begin{cases} 
> 0, & \mathbf{x} \in X^+; \\
= 0, & \mathbf{x} \in X + or X^-; \\
< 0, & \mathbf{x} \in X^-,
\end{cases}
$$

(A - 22)

with

$$
\mathbf{t} = \mathbf{D} \mathbf{K}^\top \mathbf{D} \left( \frac{1}{\mathbf{\epsilon}} + \mathbf{G} \mathbf{G}^\top \right)^{-1} \mathbf{e},
$$

(A - 23)

$$
\gamma = \mathbf{e}^\top \mathbf{D} \left( \frac{1}{\mathbf{\epsilon}} + \mathbf{G} \mathbf{G}^\top \right)^{-1} \mathbf{e},
$$

(A - 24)

and

$$
\mathbf{G} = \mathbf{D} \left[ \mathbf{K} \quad -\mathbf{e} \right].
$$

(A - 25)

Instead of $\mathbf{a}$, we have $\mathbf{K}$ in Equations A-23 and A-25, which is a Gaussian kernel function of $\mathbf{a}$ and $\mathbf{a}^T$ that has the form of:

$$
\mathbf{K}(\mathbf{a}, \mathbf{a}^T)_{ij} = \exp \left( -\sigma \| \mathbf{a}^T_i - \mathbf{a}^T_j \|^2 \right), \quad i, j \in [1, J],
$$

(A - 26)

where $\sigma$ is a scalar parameter. Finally, by replacing $\mathbf{x}^\top \mathbf{a}^T$ by its corresponding kernel expression, the decision condition can be written as:

$$
\mathbf{K}(\mathbf{x}^\top, \mathbf{a}^T) \mathbf{D} \mathbf{t} - \gamma \begin{cases} 
> 0, & \mathbf{x} \in X^+; \\
= 0, & \mathbf{x} \in X + or X^-; \\
< 0, & \mathbf{x} \in X^-.
\end{cases}
$$

(A - 27)

and

38
\[ K(x^T, a^T)_{ij} = \exp(-\sigma \|x - a^T_i\|^2), \ i \in [1,J]. \]  

(A–28)

The formulations above represent a nonlinear PSVM classifier.

To extend this binary classifier to handle multiclass classification problems, some strategies have been developed by researchers, which generally fall into three categories: “one-versus-all”, “one-versus-one” and “all together”. For \( Q \) classes, the former two strategies build a suite of binary classifiers individually: \((Q(Q - 1))/2\) for the “one-versus-one” and \( Q \) for the “one-versus-all” algorithm, and then use these classifiers to construct the final classification decision. The “all together” attempts to solve multiclass problem in one step. Hsu and Lin (2002) found “one-versus-one” method to be superior for large problems. There are two particular algorithms for “one-versus-one” strategies, namely the “Max Wins” (Kreßel, 1999) and directed acyclic graph (DAG) (Platt et al., 2000) algorithms. Both algorithms provide comparable results while surpassing the “one-versus-all” method in accuracy and computational efficiency.

Our approach uses a classification factor table to assign classes to unknown samples (Figure A4). A classification factor of an unknown sample point for a certain pilot class “X” is the normalized distance to the binary decision boundary between “X” and the other class used when generating this binary decision boundary. An example of a classification factor table is shown in Figure A4, and based on this table, the unknown sample point belongs to class “D”.
REFERENCES


Platt, J. C., N. Cristianini, and J. Shawe-Taylor, 2000, Large margin DAGs for multiclass classification: NIPS, 12, 547-553.


Roy, A., B.L. Dowdell, and K.J. Marfurt, 2013, Characterizing a Mississippian tripolitic chert reservoir using 3D unsupervised and supervised multiatribute seismic facies analysis: An example from Osage County, Oklahoma: Interpretation, 1, SB109-SB124.


Zhang, B., T. Zhao, X. Jin, and K. J. Marfurt, 2015, Brittleness evaluation of resource plays by integrating petrophysical and seismic data analysis (accepted by Interpretation).


LIST OF FIGURE CAPTIONS

Figure 1. Classification as applied to the interpretation of seismic facies (Modified from Duda et al., 2000).

Figure 2. Cartoon illustration of a k-means classification of 3 clusters. (a) Select 3 random or equally spaced, but distinct seed points, which serve as the initial estimate of the vector means of each cluster. Next, compute the Mahalanobis distance between each data vector and each cluster mean. Then color code or otherwise label each data vector to belong to the cluster that has the smallest Mahalanobis distance. (b) Recompute the means of each cluster from the previously defined data vectors. (c) Recalculate the Mahalanobis distance from each vector to the new cluster means. Assign each vector to the cluster that has the smallest distance. (d) The process continues until the changes in means converge to their final locations. If we now add a new (yellow) point, we will use a Bayesian classifier to determine into which cluster it falls (Figure courtesy of Scott Pickford).

Figure 3. (a) A distribution of data points in 3-dimensional attribute space. The statistics of this distribution can be defined by the covariance matrix. (b) k-means will cluster data into a user-defined number of distributions (4 in this example) based on Mahalanobis distance measure. (c) The plane that best fits these data is defined by the first two eigenvectors of the covariance matrix. The projection of the 3D data onto this plane provides the first two principle components of the data as well as the initial model for both our SOM and GTM algorithms. (d) SOM and GTM deform the initial 2D plane into a 2D “manifold” that better fits the data. Each point on the deformed 2D manifold is in turn mapped to a 2D rectangular “latent” space. Clusters are color-coded or interactively defined on this latent space.
Figure 4. (a) $K$ grid points $u_k$ defined on a $L$-dimensional latent space grid are mapped to $K$ grid points $m_k$ lying on a non-Euclidean manifold in $N$-dimensional data space. In this paper, $L=2$ and will be mapped against a 2-dimensional color bar. The Gaussian mapping functions are initialized to be equally spaced on the plane defined by the first two eigenvectors. (b) Schematic showing the training of the latent space grid points to a data vector $a_j$ lying near the GTM manifold using an expectation maximization algorithm. The posterior probability of each data vector is calculated for all Gaussian centroids points $m_k$ and are assigned to the respective latent space grid points $u_k$. Grid points with high probabilities are displayed as bright colors. All variables are discussed in Appendix.

Figure 5. Cartoon of a linear SVM classifier separating black from white data vectors. The two dashed lines are the margins defined by support vector data points. The red decision boundary falls midway between the margins, separating the two clusters. If the data clusters overlap, no margins can be drawn. In this situation the data vectors will be mapped to a higher dimensional space where they can be separated.

Figure 6. Cartoon describing semi-supervised learning. Blue squares and red triangles indicate two different interpreter defined classes. Black dots indicate unclassified points. In semi-supervised learning, unclassified data vectors 1 and 2 are classified to be class “A” while data vector 3 is classified to be class “B” during the training process.
Figure 7. (a) Cartoon showing a two-class PSVM in 2D space. Classes “A” and “B” are approximated by two parallel lines that have been pushed as far apart as possible forming the cluster “margins”. The red decision-boundary lies midway between the two margins. Maximizing the margin is equivalent to minimizing $\left(\omega^T \omega + \gamma^2\right)^{1/2}$. (b) A two-class PSVM in 3D space. In this case the decision-boundary and margins are 2D planes.

Figure 8. Cartoon show how one SVM can map two linearly inseparable problem into a higher dimensional space in which they can be separated. (a) Circular classes “A” and “B” in a 2D space cannot be separated by a linear decision-boundary (line). (b) Mapping the same data into a higher 3-dimensional “feature” space using the given projection. This transformation allows the two classes to be separated by the green plane.

Figure 9. A map showing the location of the 3D seismic survey acquired over the Canterbury Basin, offshore New Zealand. The black rectangle denotes the limits of the Waka-3D survey, while the smaller red rectangle denotes the part of the survey shown in subsequent figures. Colors represent the relative depth of the current seafloor, warm being shallower and cold being deeper. Current seafloor canyons are delineated in this map, which are good analogs for the paleocanyons in Cretaceous and Tertiary ages (Modified from Mitchell and Neil, 2012).

Figure 10. Time slice at $t=1.88$ s through the seismic amplitude volume. White arrows indicate potential channel/ canyon features. The yellow arrow indicates a high amplitude feature. Red arrows indicate relatively low energy, gently dipping area. AA’ denotes a cross section shown in Figure 14.
**Figure 11.** Time slice at $t=1.88$ s through peak spectral frequency co-rendered with peak spectral magnitude that emphasizes the relative thickness and reflectivity of the turbidite system and surrounding slope fan sediments into which it was incised. The two attributes are computed using a continuous wavelet transform algorithm. The edges of the channels are delineated by Sobel filter similarity.

**Figure 12.** Time slice at $t=1.88$ s through the GLCM homogeneity attribute co-rendered with Sobel filter similarity. Bright colors highlights areas with potential fan sand deposits.

**Figure 13.** Time slice at $t=1.88$ s through the co-rendered shape index, curvedness, and Sobel filter similarity. The shape index highlights incisement, channel flanks, and levees providing an excellent image for interactive interpreter-driven classification. However, the shape index dominates the unsupervised classifications, highlighting valley and ridge features and minimizing more planar features of interest in the survey.

**Figure 14.** Vertical slices along line AA’ (location shown in Figure 10) through a) seismic amplitude, b) seismic amplitude co-rendered with peak spectral magnitude and peak spectral frequency, c) seismic amplitude co-rendered with GLCM homogeneity, and d) seismic amplitude co-rendered with shape index and curvedness. White arrows indicate incised channel and canyon features. The yellow arrow indicates at a high amplitude reflector. Red arrows indicate relatively low amplitude, gently dipping areas.

**Figure 15.** Time slice at $t=1.88$ s through $k$-means classification volume with $K=16$. White arrows indicate channel-like features. Yellow arrows indicate high amplitude overbank deposits. Red arrows indicate possible slope fans. The edges of the channels are delineated by Sobel filter similarity.
**Figure 16.** Time slice at \( t=1.88 \) s through \( k \)-means classification volume with \( K=256 \). The classification result follows the same pattern as \( K=16 \) but is more chaotic since the classes are computed independently and are not constrained to fall on a lower dimensional manifold. Note the similarity between clusters of high amplitude overbank (yellow arrows) and slope fan deposits (red arrows) which were separable in Figure 15.

**Figure 17.** Time slice at \( t=1.88 \) s of the first two principle components plotted against a 2D color bar. These two principal components serve as the initial model for both the SOM and GTM images that follow. With each iteration, the SOM and GTM manifolds will deform to better fit the natural clusters in the input data.

**Figure 18.** Time slice at \( t=1.88 \) s through an SOM classification volume using 256 clusters. White arrows indicate channel-like features. Combined with vertical sections through seismic amplitude, we interpret overbank deposits (yellow arrows), crevasse splays (orange arrows), and slope fan deposits (red arrows). The data are mapped to a 2D manifold initialized by first two principle components and are somewhat more organized than the \( k \)-means image shown in the previous figures.

**Figure 19.** Time slice at \( t=1.88 \) s through crossplotting GTM projection 1 and 2 using a 2D colorbar. White arrows indicate channel-like features, yellow arrows overbank deposits, and red arrows slope fan deposits. The blue arrow indicates a braided channel system that can be seen on PCA but cannot be identified from \( k \)-means or SOM classification maps. The color indicates the location of the mean probability of each data vector mapped into the 2D latent space.
**Figure 20.** The same time slice through the GTM projections shown in the previous image but now displayed as four seismic facies. To do so, we first create two GTM “components” aligned with the original first two principal components. We then pick four colored polygons representing four seismic facies on the histogram generated using a commercial crossplot tool. This histogram is a map of the GTM posterior probability distribution in the latent space. The yellow polygon represents overbank deposits, the blue polygon channels/canyons, the green polygon slope fan deposits, and the red polygon “everything else”.

**Figure 21.** Time slice at $t=1.88 \, s$ through co-rendered peak spectral frequency, peak spectral magnitude, and Sobel filter similarity volumes. Seed points (training data) are shown with colors for the picked four facies, blue indicating multistoried channels, yellow point bars and crevasse splays, red channel flanks, and green slope fans. Attribute vectors at these seed points are used as training data in supervised classification.

**Figure 22.** PNN errors through the training epochs. The neural network reaches its best performance at epoch 42.

**Figure 23.** Confusion tables for the same PNN shown in Figure 21. From these tables we find the training correctness to be 90%, the testing and cross-validation correctness to be 86% and 91%, warranting a reliable prediction.

**Figure 24.** Time slice at $t=1.88 \, s$ through the ANN classification result. White arrows indicate channels/canyons. Yellow arrows indicate point bars and crevasse splays.
**Figure 25.** Time slice at $t=1.88 \, s$ through SVM classification result. White arrows indicate more correctly classified slope fans. Yellow arrow indicates crevasse splays. Red arrows show the misclassifications due to possible acquisition footprint.

**Figure 26.** Time slice at $t=1.88 \, s$ through inline dip component of reflector dip. Inline dip magnitude provides a photo-like image of the paleocanyons.

**Figure A1.** Self-organizing maps (SOM) workflow.

**Figure A2.** Generative topographic mapping (GTM) workflow.

**Figure A3.** Artificial neural network (ANN) workflow.

**Figure A4.** Proximal support vector machine (PSVM) workflow.
LIST OF TABLE CAPTIONS

Table 1. Attribute expressions of seismic facies.

Table 2. Classification settings and runtimes.

Table A1. List of shared mathematical symbols.
Conventional interpretation of geologic formations; geometric attribute delineation of channel-levee complexes, fault blocks, ...

Reduce seismic wiggle samples to a few key measures (AI, spectral components, curvature, ...)

Interpreter-driven classification using geologic models

Interpreter evaluation of alternative working hypotheses

Validation. Do we need to modify selected attributes?
Fig. 3
Fig. 4

a) L-dimensional Latent Space

K grid points \( u_k \)

\[ u_k \rightarrow m_k \]

\( p(x_n | k, W, \beta) \)

N-dimensional Data Space

\[ m_k \]

2D manifold \( \mathcal{S} \)

\[ a_1 \]

\[ a_2 \]

\[ a_3 \]

J basis function centers

b) Probability distribution to fit the 2D manifold \( \mathcal{S} \) to the data vector \( x_n \)

\[ p(x_n | k, W, \beta) \]

Posterior probability projection of \( x_n \) on the 2D latent space

\[ R_{nk} \]

Posterior Probability

\[ 1/\beta \]

2D manifold \( \mathcal{S} \)

Probability

0

1
Fig. 6

The diagram illustrates a scatter plot with three classes: Class A, Class B, and Unknown. The axes are labeled as $a_1$ and $a_2$. The data points are colored and labeled accordingly:

- Class A: Blue squares
- Class B: Red triangles
- Unknown: Black circles

The points are distributed across the plane, with some points circled for emphasis.
Fig. 7

(a) Decision-boundary $\mathbf{a'} \cdot \mathbf{\omega} - \gamma = 0$
- Plus-plane $\mathbf{a'} \cdot \mathbf{\omega} - \gamma = 1$
- Minus-plane $\mathbf{a'} \cdot \mathbf{\omega} - \gamma = -1$

(b) Three-dimensional visualization
- Class A: Black circles
- Class B: White circles
- Decision-boundary and Margin
- Plus-plane and Margin
- Minus-plane and Margin
Denotes “A”

\[ x^2 + y^2 = 1 \]

Denotes “B”

\[ x^2 + y^2 = 2 \]

\((x, y)\)\[\rightarrow\] \((x, y, x^2 + y^2)\)
Fig. 9
Fig. 12

GLCM homogeneity

$0.7$

$0.1$

$1$

$0$

$0.3$

$0\%$

$100\%$

$N$

$5\text{ km}$

$0$

$\text{Similarity}$

$\text{Opacity}$

$\text{Histogram}$

$t = 1.88\text{ s}$

$\text{Inline}$

$\text{Crossline}$
Fig. 14
Fig. 15

$k$-means clusters

$t = 1.88\ s$
Fig. 16

Similarity

0 100%

Opacity

$k$-means clusters

$t = 1.88 \text{ s}$
Fig. 19

- 2D histogram
- GTM latent axis 1
- GTM latent axis 2
- 0.3
- 100%
- Incline
- Crossline
- t = 1.88 s
- 2D colorbar
- Similarity
- Opacity
- GTM latent axis 1
- GTM latent axis 2
Fig. 21

<table>
<thead>
<tr>
<th>Facies index</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue</td>
<td>Channel stories</td>
</tr>
<tr>
<td>Yellow</td>
<td>Point bars and crevasse splays</td>
</tr>
<tr>
<td>Red</td>
<td>Channel flanks</td>
</tr>
<tr>
<td>Green</td>
<td>Slope fans</td>
</tr>
</tbody>
</table>

$F = 1.88$ s
Best Validation Performance is 0.14482 at epoch 42
**Training Confusion Matrix**

<table>
<thead>
<tr>
<th>Output Class</th>
<th>Target Class 1</th>
<th>Target Class 2</th>
<th>Target Class 3</th>
<th>Target Class 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>45 (14.8%)</td>
<td>1 (0.3%)</td>
<td>4 (1.3%)</td>
<td>5 (1.6%)</td>
</tr>
<tr>
<td>2</td>
<td>0 (0.3%)</td>
<td>56 (18.4%)</td>
<td>0 (0.0%)</td>
<td>0 (0.0%)</td>
</tr>
<tr>
<td>3</td>
<td>16 (5.2%)</td>
<td>0 (0.0%)</td>
<td>66 (21.6%)</td>
<td>3 (1.0%)</td>
</tr>
<tr>
<td>4</td>
<td>0 (0.0%)</td>
<td>0 (0.0%)</td>
<td>1 (0.3%)</td>
<td>107 (35.1%)</td>
</tr>
<tr>
<td></td>
<td>72.6%</td>
<td>98.2%</td>
<td>93.0%</td>
<td>89.8%</td>
</tr>
</tbody>
</table>

**Validation Confusion Matrix**

<table>
<thead>
<tr>
<th>Output Class</th>
<th>Target Class 1</th>
<th>Target Class 2</th>
<th>Target Class 3</th>
<th>Target Class 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9 (13.6%)</td>
<td>1 (1.5%)</td>
<td>1 (1.5%)</td>
<td>2 (3.0%)</td>
</tr>
<tr>
<td>2</td>
<td>0 (0.0%)</td>
<td>17 (25.8%)</td>
<td>0 (0.0%)</td>
<td>0 (0.0%)</td>
</tr>
<tr>
<td>3</td>
<td>2 (3.0%)</td>
<td>0 (0.0%)</td>
<td>11 (16.7%)</td>
<td>0 (0.0%)</td>
</tr>
<tr>
<td>4</td>
<td>0 (0.0%)</td>
<td>0 (0.0%)</td>
<td>0 (0.0%)</td>
<td>23 (34.8%)</td>
</tr>
<tr>
<td></td>
<td>81.8%</td>
<td>94.4%</td>
<td>91.7%</td>
<td>90.9%</td>
</tr>
</tbody>
</table>

**Test Confusion Matrix**

<table>
<thead>
<tr>
<th>Output Class</th>
<th>Target Class 1</th>
<th>Target Class 2</th>
<th>Target Class 3</th>
<th>Target Class 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11 (16.7%)</td>
<td>1 (1.5%)</td>
<td>2 (3.0%)</td>
<td>2 (3.0%)</td>
</tr>
<tr>
<td>2</td>
<td>0 (0.0%)</td>
<td>16 (24.2%)</td>
<td>0 (0.0%)</td>
<td>0 (0.0%)</td>
</tr>
<tr>
<td>3</td>
<td>1 (1.5%)</td>
<td>0 (0.0%)</td>
<td>11 (16.7%)</td>
<td>2 (3.0%)</td>
</tr>
<tr>
<td>4</td>
<td>1 (1.5%)</td>
<td>0 (0.0%)</td>
<td>0 (0.0%)</td>
<td>19 (28.8%)</td>
</tr>
<tr>
<td></td>
<td>84.6%</td>
<td>94.1%</td>
<td>84.6%</td>
<td>86.4%</td>
</tr>
</tbody>
</table>

**All Confusion Matrix**

<table>
<thead>
<tr>
<th>Output Class</th>
<th>Target Class 1</th>
<th>Target Class 2</th>
<th>Target Class 3</th>
<th>Target Class 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>65 (14.9%)</td>
<td>3 (0.7%)</td>
<td>7 (1.6%)</td>
<td>9 (2.1%)</td>
</tr>
<tr>
<td>2</td>
<td>1 (0.2%)</td>
<td>89 (20.4%)</td>
<td>0 (0.0%)</td>
<td>0 (0.0%)</td>
</tr>
<tr>
<td>3</td>
<td>19 (4.3%)</td>
<td>0 (0.0%)</td>
<td>88 (20.1%)</td>
<td>5 (1.1%)</td>
</tr>
<tr>
<td>4</td>
<td>1 (0.2%)</td>
<td>0 (0.0%)</td>
<td>1 (0.2%)</td>
<td>149 (34.1%)</td>
</tr>
<tr>
<td></td>
<td>75.6%</td>
<td>96.7%</td>
<td>91.7%</td>
<td>91.4%</td>
</tr>
</tbody>
</table>

**Target Class**

1 - **Action 1**
2 - **Action 2**
3 - **Action 3**
4 - **Action 4**
Fig. 25

Similarity

0 100%

Opacity

Facies index

<table>
<thead>
<tr>
<th>Facies</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue</td>
<td>Channel stories</td>
</tr>
<tr>
<td>Yellow</td>
<td>Point bars and crevasse splays</td>
</tr>
<tr>
<td>Red</td>
<td>Channel flanks</td>
</tr>
<tr>
<td>Green</td>
<td>Slope fans</td>
</tr>
</tbody>
</table>

$t = 1.88$ s
Fig. A1

1. **Input multi-attribute data** $a_j$
2. **Z-score**
3. **Normalize data vector** $a_j$
4. **Randomly select the data vector** $a_j$, find the Best match with the prototype vectors $m_k$.
5. **Deform neighboring prototype vectors** $m_k$.
   - **$r < \sigma$**
     - **Yes**
       - **More iterations?**
         - **Yes**
           - **Training Loop**
         - **No**
           - **Sammon projection**
       - **No**
         - **Shrink neighborhood** $\sigma$ and run training loop
8. **Initialize manifold prototype vectors** $m_k$ to span the first two principal components in the data-space.
9. **Find nearest prototype vector** $m_k$ to each data point $a_j$. Assign $a_j$ to class $k$.
10. **Project final** $m_k$ against onto the 2D latent space grid $u_k$.
11. **Color the projected points** $u_k$ with 1D or 2D colorbar and color-code corresponding data points.
12. **Final manifold Prototype Vectors** $m_k$.
13. **Final Seismic facies volume**
Input multi-attribute data $a_j$

Z-score

Initialize the 2D latent space grid $u_k$ and interpolation function $\Phi(u_k)$

Initialize manifold $m_k$ to span the first two principal components in the data-space

Initialize $W$ and $\beta$

Map the latent space vectors $u_k$ to data space manifold vectors $m_k = W\Phi(u_k)$

Compute Gaussian with centers at each $m_k$

Calculate the posterior probabilities $R_{kj}$

$\Delta\beta$ small?

No

Updated $W$ and $\beta$

Calculate the new posterior responsibility, $R_{kj}$

Update $W$ and $\beta$

Yes

Final manifold $m_k$

Project the final mean posterior responsibilities $<R_{kj}>$ onto the the 2D Latent space using crossplot software

Final Seismic facies volume

Draw polygons around desired clusters and color-code corresponding data points

Fig. A2
Classified interpreter seed picks

Input multi-attribute data \( \mathbf{a}_j \)

Attribute extraction

Training data (classed attribute vectors)

Form perceptrons

\[
y_j = \sum_{n=0}^{N} w_n a_{nj}\\
r_j = \frac{1}{1 + e^{-y_j}}
\]

Apply perceptrons

\[
y_j = \sum_{n=0}^{N} w_n a_{nj}\\
r_j = \frac{1}{1 + e^{-y_j}}
\]

Output classes

Validation data (classed attribute vectors)

Validation OK?

Yes

Accept Classification

No

Reject Classification

Reselect input attributes and seed picks

Update \( \mathbf{W} \)

Training error small?

No

Final weights, \( \mathbf{W} \)

Yes

Accept Classification

Reselect input attributes and seed picks

Fig. A3
Example of a classification factor table

Evaluate the binary PSVM classification factor, \( f(q;p) \), of the current pilot class, \( p \), against all remaining active classes, \( q \).

All \( f(q;p) > 0 \)?

Yes

Assign the current pilot class, \( p \), to current voxel

No

Set old pilot, \( p \), to be inactive
Set new \( p \) to be that \( q \) that gives \( \min[f(q;p)] \)

For training samples of \( Q \) classes, generate \( Q(Q-1)/2 \) binary classifiers for every two classes

Initialize all classes to be active

Choose initial pilot class, \( p \)

Input multi-attribute data \( a_j \)
<table>
<thead>
<tr>
<th>Facies</th>
<th>Appearance to Interpreter</th>
<th>Attribute Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Levee</td>
<td>Structurally high</td>
<td>Stronger dome or ridge shape structural components</td>
</tr>
<tr>
<td></td>
<td>locally continuous</td>
<td>Higher GLCM homogeneity; lower GLCM entropy</td>
</tr>
<tr>
<td></td>
<td>Higher amplitude</td>
<td>Dome or ridge shape component</td>
</tr>
<tr>
<td></td>
<td>Possibly thicker</td>
<td>Lower peak spectral frequency</td>
</tr>
<tr>
<td>Channel thalwegs</td>
<td>Shale-filled with negative compaction</td>
<td>Stronger bowl or valley shape structural components; higher peak spectral frequency</td>
</tr>
<tr>
<td></td>
<td>Sand-filled with positive compaction</td>
<td>Stronger dome or ridge shape structural components; lower peak spectral frequency</td>
</tr>
<tr>
<td>Channel flanks</td>
<td>Onlap onto incision, canyon edges</td>
<td>Higher reflector convergence magnitude</td>
</tr>
<tr>
<td>Gas-charged sands</td>
<td>High amplitude, continuous reflections</td>
<td>Higher GLCM homogeneity; lower GLCM entropy; high high peak magnitude</td>
</tr>
<tr>
<td>Incised floodplain</td>
<td>Erosional truncation</td>
<td>Higher reflector convergence magnitude, Higher curvedness</td>
</tr>
<tr>
<td>Floodplain</td>
<td>Lower amplitude</td>
<td>Lower spectral magnitude</td>
</tr>
<tr>
<td></td>
<td>Higher frequency</td>
<td>Higher peak spectral frequency</td>
</tr>
<tr>
<td></td>
<td>Continuous</td>
<td>Higher GLCM homogeneity; lower GLCM entropy</td>
</tr>
<tr>
<td></td>
<td>Near planar events</td>
<td>Lower amplitude structural shape components; lower reflector convergence magnitude</td>
</tr>
<tr>
<td>Slumps</td>
<td>chaotic reflectivity</td>
<td>Higher reflector convergence magnitude; higher spectral frequency; lower GLCM homogeneity; higher GLCM entropy</td>
</tr>
<tr>
<td>Algorithm</td>
<td>Number of classes</td>
<td>MPI processors*</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>k-means</td>
<td>16</td>
<td>50</td>
</tr>
<tr>
<td>k-means</td>
<td>256</td>
<td>50</td>
</tr>
<tr>
<td>SOM</td>
<td>256</td>
<td>1</td>
</tr>
<tr>
<td>GTM</td>
<td>-</td>
<td>50</td>
</tr>
<tr>
<td>ANN†</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>SVM</td>
<td>4</td>
<td>50</td>
</tr>
</tbody>
</table>

*SOM is not run under MPI in our implementation. ANN is run using MATLAB® and is not under MPI. All other three are run under MPI when applying the model to the entire dataset.

†ANN is implemented using MATLAB® toolbox.
<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n, N$</td>
<td>attribute index and number of attributes</td>
</tr>
<tr>
<td>$j, J$</td>
<td>voxel (attribute vector) index and number of voxels</td>
</tr>
<tr>
<td>$k, K$</td>
<td>manifold index and number of grid points</td>
</tr>
<tr>
<td>$a_j$</td>
<td>the $j^{th}$ attribute data vector</td>
</tr>
<tr>
<td>$p$</td>
<td>matrix of principle components</td>
</tr>
<tr>
<td>$C$</td>
<td>attribute covariance matrix</td>
</tr>
<tr>
<td>$\mu_n$</td>
<td>mean of the $n^{th}$ attribute</td>
</tr>
<tr>
<td>$\lambda_m \mathbf{v}_m$</td>
<td>the $m^{th}$ eigenvalue and eigenvector pair</td>
</tr>
<tr>
<td>$m_k$</td>
<td>the $k^{th}$ grid point lying on the manifold (prototype vector for SOM, or Gaussian center for GTM)</td>
</tr>
<tr>
<td>$u_k$</td>
<td>the $k^{th}$ grid point lying on the latent space</td>
</tr>
<tr>
<td>$r_{jk}$</td>
<td>the Mahalanobis distance between the $j^{th}$ data vector and the $k^{th}$ cluster center or manifold grid point</td>
</tr>
<tr>
<td>$I$</td>
<td>Identity matrix of dimension defined in the text</td>
</tr>
</tbody>
</table>