Statistical imaging of faults in 3D seismic volumes using a machine learning approach
Antoine Guitton*, Hua Wang† and Whitney Trainor-Guitton*
* Geophysics Department, Colorado School of Mines, Golden, CO, USA
† Computer Science Department, Colorado School of Mines, Golden, CO, USA

SUMMARY

Faults are highlighted in 3D seismic volumes with a supervised machine learning algorithm. We label the faults using an automatic fault-picking method developed by Hale (2013). We build feature vectors for the training and classification steps using two popular techniques in object recognition algorithms called Histograms of Oriented Gradients (HOG) and Scale Invariant Feature Transforms (SIFT). We train and classify the seismic data using a Support Vector Machine classifier with Gaussian kernels. Using both SIFT and HOG features together reduces the false positive rate, delivering better fault images. Our approach is able to predict faults in both synthetic and field data cubes quite well even when mislabeled data are used for training.

INTRODUCTION

Interpretation tasks requiring human input are by nature slow, expensive, and not repeatable. In exploration geophysics, the interpretation of large seismic volumes remains one of the most labor-intensive tasks. Enormous progress has been achieved to facilitate these tedious steps. The automatic interpretation of 3D volumes based on semblance analysis, phase unwrapping (Stark, 2004) or local dip integration (Lomask et al., 2006) has revolutionized the way interpretation can be done: using deterministic approaches and simple differential operators (Hale, 2013), codes can reliably extract the information of interest in large volumes. Contrary to humans, interpretation tasks done by computers are fast, cheap and repeatable. In addition, whereas humans are particularly efficient at finding features in 2D only, properly written softwares can work in any dimension.

Here, we take a statistical approach to the interpretation of faults and use machine learning algorithms (MLAs) to identify them. Our goal is not to pick fault surfaces, but to provide maps of possible fault locations (also known as fault imaging or highlighting). Our method follows a supervised learning concept. In a nutshell, for input, we have a set of features that we can arrange in a vector form. We write \( x_j \) the value of the feature \( i \) for data point \( j \). For each point, we have a feature vector \( x_j \) and we want to find a function \( f(x_j) \) such that \( y_j = f(x_j) \). Putting all the feature vectors into one matrix \( X \) and all outcomes in one column vector \( y \) we seek a function \( f(X) \) such that

\[
y = f(X)
\]

(1)

Figure 1: A synthetic training seismic data volume with its interpreted faults.

CONCEPTS OF MACHINE LEARNING

In this section, we introduce basic notations and concepts to better understand machine learning algorithms. Machine learning can be applied to labeled, semi-labeled, or unlabeled data. In the labeled case, all samples used for the training have a quantitative or qualitative value.

For our fault highlighting method, we follow a supervised learning approach. Our outcome can take only two qualitative values, or classes: \( y = \{ \text{fault, no fault} \} \). For convenience, we prefer working with values \( y = \{1,0\} \). Now from the seismic data, we can extract at each point some features. These features can be, for instance, gradients of the image, or semblances, or simply pixel values. A data point can be a collection of pixels in a small neighborhood, or patch. So at each point, we have a set of features that we can arrange in a vector form. We write \( x_j \) the value of the feature \( i \) for data point \( j \). For each point, we have a feature vector \( x_j \) and we want to find a function \( f(x_j) \) such that \( y_j = f(x_j) \).

Support vector machine classifier

Support vector machine (SVM) classifiers are very popular to build an optimal hyperplane separating two classes (Hastie et al., 2001). It can also discriminate between non-linearly separable classes using so-called “kernels”, or similarity functions. It is not the goal of this paper to describe the SVM classifier in details, but merely expose its interesting properties. SVM classifiers are called wide-margin classifiers, meaning that they find the boundaries between two classes that maximize the distance between feature vectors and decision boundaries. The vectors defining the position of the margin (and on the margin) are called the support vectors.

The penalty, or loss, function associated with the SVM...
classifier takes the form
\[ g(\theta) = \sum_j y_j H_1(f(x_j)) + (1 - y_j) H_0(1 - f(x_j)) + \lambda \|\theta\|_2^2 \]  

(2)

where \( H_1 \) and \( H_0 \) are hinge loss functions defined as
\[ H_1(x) = \max\{0, 1 - x\} \]
\[ H_0(x) = \max\{0, 1 + x\} \]  

(3)

With this definition, points within a class have a weight of zero and do not contribute to the total loss. Only points on the margin or between the two margins count. In addition, the contribution of these points is linear adding robustness to outliers.

To accommodate non-linearly separable classes, the notion of kernels enters into the definition of the hypothesis \( f \) as follows (James et al., 2013)
\[ f(x_j) = \theta^T h(x_j) \]  

(4)

and
\[ h(x_j) = \sum_{i=1}^{N} K(x_j, x_i) \]  

(5)

where \( K(x, x') \) is called the kernel and should be a positive semi-definite function and \( N \) is the number of training points. Kernels help defining a metric relating two vectors. They are also called similarity functions and create a local weight around each point. In this paper, we use the radial/Gaussian basis function. The classification is done by estimating \( f \) in equation 4:
\[
\begin{cases} 
  y_j = 1 & \text{if } f(x_j) > 0 \\
  y_j = 0 & \text{otherwise}
\end{cases}
\]  

(6)

The SVM classifier doesn’t provide classes probabilities. Training an SVM classifier requires parameters tuning. In this work, the parameters are selected with 5-fold cross-validation (James et al., 2013).

FEATURES COMPUTATION

The most simple feature is the seismic data itself, using amplitude information only. Unfortunately, amplitude is a very poor predictor of faults in seismic data. Here, we opted for two classes of features that are widely used in computer vision for object recognition. One is called Scale Invariant Feature Transform (SIFT) (Lowe, 1999), and the other one is called Histogram of Oriented Gradients (HOG) (Dalal and Triggs, 2005).

Histogram of Oriented Gradients

The computation of HOG features is quite simple (Dalal and Triggs, 2005). First, an image is decomposed into cells and blocks. Cells are usually half the size (in number of pixels) of blocks. Blocks are mostly used for normalization purposes of the histograms. In general, but not always, cells are 8x8 and blocks 16x16. The first step consists in computing vertical \( \nabla_z \) and horizontal \( \nabla_x \) gradients using centered 1D derivative operators \([-1, 0, 1]\) at each location in the image. From

Figure 2: (a) Test data. (b) True fault locations after reassembling the patches. (b) predicted faults locations when HOG features are used only. (c) Predicted fault locations when HOG and SIFT features are used. The prediction is cleaner than in Figure 2c with fewer mis-interpreted faults on the edges of the cube.
Figure 3: (a) Test data. (b) Fault images picked automatically. Predicted faults using (c) HOG features and (d) HOG+SIFT features. The color scale goes from 0 (green) to 1 (red) and could be interpreted as a fault probability map. Smoothing improves temporal and spatial continuity.

these gradients, signed (between $0^\circ - 360^\circ$) and unsigned (between $0^\circ - 180^\circ$) angles are computed. Next, histograms of angles are computed for each cell. In practice, nine orientations bins are usually estimated, where each angle is linearly interpolated between neighboring bins. Once histograms have been estimated, a normalization step follows. The normalization compensates for local variations in amplitudes due to illumination effects, noise, geology etc... The normalization is done for each block (i.e., groups of cells) where blocks are overlapped.

Scale Invariant Feature Transform

SIFT is another popular descriptor for image classification. The goal of SIFT is to match features across different images. The power of SIFT is that it can match features between images that are rotated, scaled, illuminated or viewed from different angles.

The first step of SIFT consists in finding keypoints in the image. These keypoints are supposed to represent prominent features in the image. To do this, the image is first transformed into a scale space: the image is simultaneously blurred with different Gaussian kernels and sub-sampled. Each scale is called an “octave”. Therefore, after this first stage, SIFT creates many duplicates of the image at different scales and with different smoothing functions. Then, for each octave, two consecutive blurred images are subtracted to create a Difference of Gaussians (DoG) scale space. In the next step, local extrema are detected by looping over all pixels for each scale within an octave and comparing it with its closest 26 neighbors. The local extrema are then quadratically interpolated so that we have a value of extrema everywhere.

Once the keypoints are identified at different scales, an orientation for each keypoint is estimated. To do this, a local dip is estimated for all points surrounding a keypoint using the same formula as the one used for the HOG method. Then a histogram of all dip values is computed. The bin with the highest value is assigned as the keypoint dip. Other keypoints with the same location and scale are created for all bins within 80% of the highest value. Finally, for each keypoint 16 blocks (4×4) are defined with 16 pixels per block (4×4). Within each block, histograms of gradient magnitudes and orientations are computed and binned in an 8 bins histogram. The keypoint orientation is subtracted from all computed orientations within surrounding blocks to preserve rotation independence. The bin values are also clipped for normalization. Therefore, for each keypoint, we have a vector of 128 descriptors (16 blocks times 8 bins). For classification, the descriptor of 128 elements is used rather than its location.

SYNTHETIC DATA EXAMPLE

For the fault imaging problem, we first need to build the required sets of labeled seismic data. For this, we
build many synthetic datasets with different fault geometries and pick them using the Mines Java Toolkit (https://github.com/dhale/jtk) and the seismic image processing for geological faults software (https://github.com/dhale/ifp). We build the features vectors $x_j$ from the seismic volumes and the outcome vector $y$ from the fault picks. An example of training data set is shown in Figure 1. The HOG and SIFT features need to be estimated on small windows (or patches). Therefore, we decompose each cube into 2D slices along the crossline direction. Each slice is then decomposed into overlapping windows of 8x6 pixels, with an overlap zone of half the window length in both directions. We estimate the HOG and SIFT features from these small windows. In this parameterization, each patch becomes one training/validation point.

**Fault interpretation with HOG only**

We first train and predict using the HOG features only. After training, we use the SVM classifier to identify faults in a seismic volume not seen during training or validation (Figure 2a). Figure 2b shows the interpreted faults after reassembling the patches of the faults images in Figure 2a. By construction, the patching makes the faults wider. We consider Figure 2b to be the answer, or true prediction, of the fault locations. Figure 2c displays the locations of our predicted faults using the HOG features only. There is a very good agreement between Figures 2b and 2c. We notice some edge effects that would be mitigated with a proper taper function. Overall, the classifier was able to identify all faults properly. Given that the feature vector is computed on 2D windows only, we think that better results would be possible by estimating features in 3D. Yet, the classifier performed remarkably well.

**Fault interpretation with HOG and SIFT**

We are now combining HOG and SIFT features to train our classifier. The classification result is shown in Figure 2d and seems to indicate a better classification than when HOG features are used only. In other words, our prediction has become more conservative and has predicted fewer faults (42326 with HOG only, 36908 with HOG and SIFT), thus resulting in a cleaner image. Because of the overlap of the patches and the inherent robustness coming with it (the information for one fault is spread among many patches), missing more faults is beneficial to the overall prediction.

**FIELD DATA EXAMPLE**

For the field data example, we select a small 3D seismic volume that we divide in three parts for validation, training and testing. We follow the same procedures as with the synthetic data. However now, the window size for each patch is 8x12, as opposed to 8x6 for the synthetic dataset. Figure 3a shows the test data, not seen by the classifier, and Figure 3b displays the labeled faults using Hale’s software (and after reassembling the outcome vector patches). The labeling of the faults is not accurate everywhere. Looking at Figure 3b we see fault locations in the test data not picked by the automated fault detection software. It might also happen that picked faults are not true faults. Therefore, the training of the classifier is done with mislabeled data, where false positives and false negatives are present.

We train the SVM with the HOG features only. The outcome vector for the test data mapped back into the original dataset space is shown in Figure 3c. To increase the vertical and horizontal continuity of faults in the crossline direction, we smoothed the fault map. It is pleasing to see that the five major vertical faults between x=609 km and x=613 km are identified by the classifier. Their lateral extent in the crossline direction is also clearly predicted by the MLA. Now, we train the SVM with both HOG and SIFT features. We map the predicted outcomes back into the seismic volume in Figure 3d. We notice that the main faults are predicted correctly. We can also see that we are missing some faults, comparing with Figure 3b, but that we are also picking real faults not visible in Figure 3b. Similar to the synthetic data examples, a combination of HOG and SIFT yields a cleaner fault image.

**DISCUSSION - CONCLUSION**

From a MLA view point, we could improve our prediction by incorporating some smoothness in the predictor directly and not in a post-processing step. For this, the method of Wang et al. (2014) using spatially-temporally consistent tensors could be used. Another improvement could come by integrating all our features (SIFT and HOG) in a better way by adding a joint-structured sparsity regularization term (Wang et al., 2013). Finally, we could use a semi-supervised approach where we don’t label all faults but just a few: this would save time and might handle the mislabeling issue with field data better.

From a feature computation view point, extending the SIFT and HOG methods to 3D is the next natural step. While the labelling is done in 3D, the features are estimated in 2D planes only. By taking the SIFT and HOG features to higher dimensions, better predictions would follow. In addition, we could use other features (semblance, dip) for the classification. Additionally, we need to include more training data to have a classifier able to identify faults in many environments.

**ACKNOWLEDGMENTS**

Antoine Guitton thanks the sponsors of the Center for Wave Phenomena and the College or Earth Resource Sciences and Engineering for financial support.
REFERENCES


Hale, D., 2013, Methods to compute fault images, extract fault surfaces, and estimate fault throws from 3d seismic images: Geophysics, 78, O33–O43.


James, G., D. Witten, T. Hastie, and R. Tibshirani, 2013, An Introduction to Statistical Learning, with Applications in R: Springer.


