DOI:10.1190/segam2020-3427836.1

Automatic velocity model building with machine learning

Chaoguang Zhou and Samuel Brown, PGS*

Summary

Velocity model building for seismic imaging is commonly performed with tomography and full wave inversion (FWI). Both techniques are time consuming and need significant human intervention. Machine learning has been introduced into seismic imaging with the goal of reproduce the success earned in other fields. Due to the complexity of the earth, and the geological uniqueness of any one location, determining the appropriate training data can be challenging. Directly building a 3D velocity model by machine learning still has some way to go. Instead of letting machine learning do all the work, it may be more practical to only perform machine learning on the portion of model building that requires heavy human intervention. In this paper, we present a method that builds the velocity model automatically by combining novel machine learning with the mature velocity model building techniques.

Introduction

Seismic imaging needs a velocity model to map the recorded seismic signals to a subsurface image. Tomography and FWI are the most common techniques for building such models. Tomography iteratively updates the velocity model and human analysis and interpretation are usually needed for each update. FWI is computationally expensive and its preparation may need significant human involvement too.

In recent years, machine learning has been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs, where they have produced results comparable to and in some cases surpassing expert human performance. Inspired by such successes, there have been numerous attempts to bring machine learning into seismic imaging (Yang and Ma, 2019; Zheng et. al., 2019). Araya-Polo et. al. (2018) and Øye and Dahl (2019) tried to build a velocity model directly from seismic shot gathers using machine learning. Although they show encouraging results, the precision and resolution of the estimated velocity models are not on par with state of the art velocity model building techniques such as reflection tomography and FWI (Øye and Dahl, 2019). One of the difficulties that prevent people from reproducing the successes in other fields is the lack of training data that represents the complex earth geology.

It might be more practical to utilize machine learning to handle the steps in model building that requires heavy human intervention while taking advantage of current mature model building techniques. In reflection tomography, residual moveout (RMO) picking typically requires significant parameter tuning and even manually editing out bad RMO picks, such as multiples and refractions. The successes of machine learning classification in other fields, for example, image recognition (Krizhevsky et. al. 2012), suggest we may be able to employ machine learning to identify and remove the bad RMO picks. In the next section, we will discuss an automatic velocity model building method that uses machine learning for RMO pick recognition.

Method

Conventional reflection tomography (Zhou et. al. 2008, 2009) is a successive model building technique with multiple steps. Starting with the initial seismic migration gathers and the initial velocity model, reflection tomography loops through the following steps (Figure 1): RMO, gamma scanning and event picking that picks the valid RMOs from the current migration gathers; Update, ray-tracing and inversion that converts the picked RMOs into velocity updates; and Migration on the current velocity model. We can encapsulate all these steps into one box (Figure 1) and automate the tomographic model building process. The tomography loop becomes internal and several separate modules can become one. Migration does not usually need any parameter changes except the velocity input. As Zhou et. al. (2009) showed, the model building process starts with larger smoothing lengths to solve for larger wavelength velocity features and gradually reduces in later iterations for small velocity variations and higher resolution. Such iteration dependent parameterization can be preset and there is no need to change this during the process.

The RMO step, however, usually requires human intervention for parameter tuning and conditioning, even manual editing in some extreme cases. This not only breaks down the automation but can also incur a long turnaround time. Automatically scanned and picked RMOs normally contain some bad picks such as multiples that we do not want to use in the subsequent velocity updating. By adjusting thresholds such as semblance and gamma range, bad picks can be largely reduced but a significant amount may remain. This is largely data dependent. Harsher thresholds remove more bad RMO picks but often also eliminate good ones. Human identification through experience is currently the optimal approach, but is time consuming. With the help of machine learning, we can create a small subset of the migration gathers and scan and pick RMOs automatically by thresholding. Then we manually correct what the automatic process does wrong and the resulting data is used in a

10.1190/segam2020-3427836.1

DOI:10.1190/segam2020-3427836.1

training process. The resulting trained model is then provided to the automatic velocity model building module as a parameter and the module runs without human interaction.

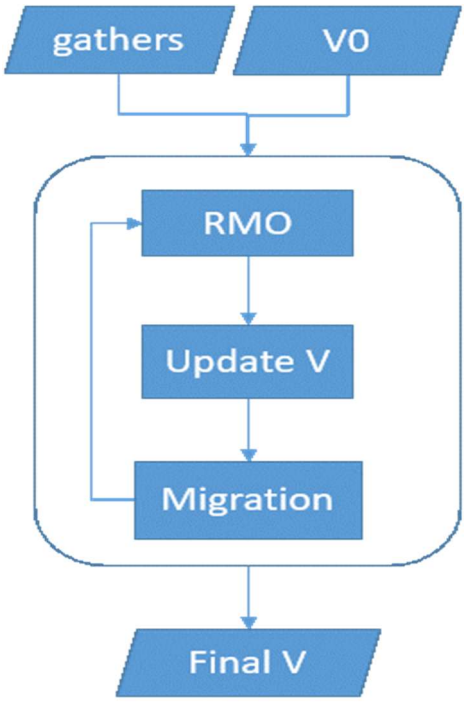

Figure 1: The automatic velocity model building diagram. All model building steps are encapsulated in a single box.

Identifying bad RMO picks with machine learning using a training process is a form of supervised classification. It is implausible to cut out the local event image for the machine to learn and identify because very often the surrounding information is also needed. For example, one may determine an event is a multiple because it differs from events around it in curvature. If we only extract an event itself, we will have difficulties to tell if it is a multiple or not, especially when there are limited offsets. Therefore, instead of extracting event images, we use following features: spatial coordinates (x, y, and z), scanned gamma value, peak semblance and local reflector dip. All feature values are normalized of the range of zero to one.

Field examples

The proposed method has been implemented with a 1D migration engine together with a 1D velocity update operator. The machine learning components were built with Python using Scikit-Learn and TensorFlow packages. Random Forest and Neural Network have been chosen as options for RMO pick classification. The automatic velocity model building has three options: Prep, Train and Production. Prep scans and automatically picks RMOs on a small subset of gathers with user defined thresholds. Then the user manually corrects the errors that exist in the output of Prep. The corrected data will be the training data that serves as the input for Train. After Train finishes training, the trained model is supplied, together with the full gathers, to Production for the velocity model building. The Production job is fully automated and contains all internal model building steps (Figure 1), of which step RMO is equipped with machine learning. Such jobs are typically parameterized to run several iterations or until a satisfactory velocity model is obtained.

A small dataset from North Sea

The first experiment was conducted on a small dataset from the North Sea. It contains only 12 inlines and 841 crosslines. The reason to test on such a small dataset is that we could quickly test some hyperparameters. In machine learning, a hyperparameter is a parameter whose value is set before the learning process begins. When we found the optimal hyperparameters values, we set them to be permanent. The migration gathers have an offset range from 596 m to 7596 m with a 200 m increment. As shown in Figure 2, the entire deep region is contaminated with multiples. We chose the first, the middle and the last inlines for training. For each training inline, we selected 42 gathers out of the total 841. The total number of gathers for training is 126. Since the thresholds defined in the job helped correctly pick the majority of events, it only took a few minutes to edit the ones that were wrongly classified. We did not have the reflector dip information and zero dips were assumed. The Production job was parameterized to run only one iteration.

First we tested the Random Forest classifier and it gave 100% accuracy on the training data and the reported importance for each feature is shown in Table 1. Features z and gamma dominate in this training experiment because almost all bad picks are in the deep region and they are all multiples that have larger gamma values. As the dips for all events are the same, they have zero importance.

Rank	Feature	Importance
1	z	0.538868
2	gamma	0.413296
3	semblance	0.026730
4	x	0.013669
5	y	0.007438
6	dip x	0.000000
7	dip y	0.000000

Table 1: Feature importance for the small North Sea dataset experiment.

DOI:10.1190/segam2020-3427836.1

The trained Random Forest model was supplied to the Production job on the whole dataset and it classified the picks as desired (Figure 2b), comparable to the training data (Figure 2a). Because all deep RMO picks were classified as bad and not used for velocity updating, the velocity update in the deep region is zero (Figure 3).

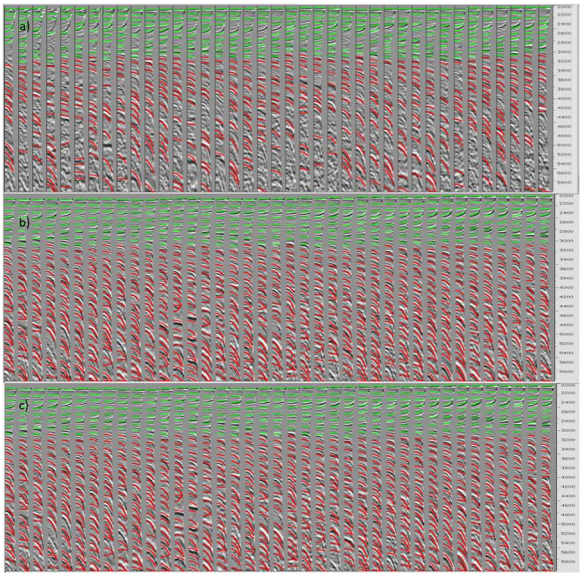

Figure 2: Manually edited training data (a) for the small North Sea dataset experiment. Classified RMO picks on the whole dataset by the Random Forest classifier (b) and the Neural Network classifier (c). Green represents good picks and red means bad picks. (a) shows gathers from three training inlines while (b) and (c) contain same gathers from one inline.

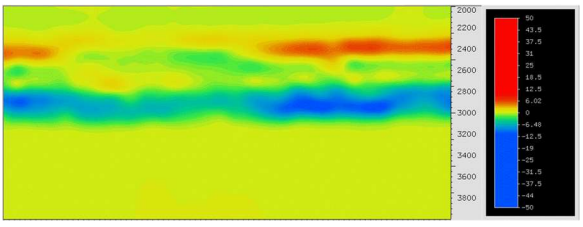

Figure 3: Velocity update for the small North Sea dataset. RMO picking used the Random Forest classifier.

The same training data was also fed to the Neural Network classifier. There are many hyperparameters and we tested the number of layers, epochs and different activations. We found a 2-layer network yielded similar result with networks of 3, 4, and 5 layers and 2000 epochs was sufficient. We also chose relu as the activation for the first layer and softmax for the second. One of the advantages of softmax is that it gives the probability for each category, which gives the user a chance to make the final decision. For example, if a RMO pick has a 51% probability to be a good one and 49% chance to be a bad one, the user may decide not to use it for velocity updating. Adam was chosen as the optimizer and other optimizers were not tested. The Neural Network training reported an accuracy of 99% on the training data. The classified RMO picks (Figure 2c) are similar to the result from the Random Forest classifier (Figure 2b), which in turn resulted in a similar velocity update.

North Sea example 2

The implemented automatic model building method has also been apllied on another larger North Sea dataset. The initial migration gathers have dimensions of 721 inlines, 1057 crosslines and 40 offsets. As shown in Figure 4, the shallow portion of the gathers has mostly primary events, which is similar to the previous example. However, in the deeper data, there are a lot of multiples, but some primairies can be identified. We also used zeros for all reflector dips since such information was not available.

Rank	Feature	Importance
1	z	0.430944
2	gamma	0.277258
3	semblance	0.171127
4	x	0.100172
5	y	0.020500
6	dip x	0.000000
7	dip y	0.000000

Table 2: Feature importance for the larger North Sea dataset experiment.

As the first example, we chose a coarse grid of 3 inlines and 80 crosslines for the training data. Due to the complexity of the RMO, manually correcting the classifications by thresholding took about 30 minutes. An experienced imager would have done the work more efficiently though. The Production job was parameterized to run 5 iterations on the whole dataset. Again, the Random Forest classifier achieved a 100 % accuracy on the training data. However, some changes were seen in the reported feature importance (Table 2). Although z and gamma still rank 1st and 2nd, but they are not as dominant as in previous example. The semblance and coordinates x and y also played siginificant roles.

As shown in Figure 4b, the Random Forest classifier did a reasonable job and most RMO picks were correctly classified. After 5 iterations of automatic updating, the Production run of the Random Forest classifier greatly flattened the seismic gathers (Figure 5b) compared with the

10.1190/segam2020-3427836.1

DOI:10.1190/segam2020-3427836.1

initial migration gathers (Figure 5a). The resulting velocity update (Figure 6) is consistent with the gather changes.

When the same training data was provided to the Neural Network classifier, it reported a 86 % accuracy on the training data, significantly lower than the Random Forest classifier. With the Neural Network trained model, the classifier performed poorly (Figure 4c). It did a fair job in the shallow region but struggled in the deep by classifying most the deep picks as bad ones. This Production run then was abandoned. The inferior perfomance suggests the hyperparameters based upon the experiments on the smaller dataset are not transferable for this dataset and further tuning is needed to obtain a new set of optimal hyperparameter values.

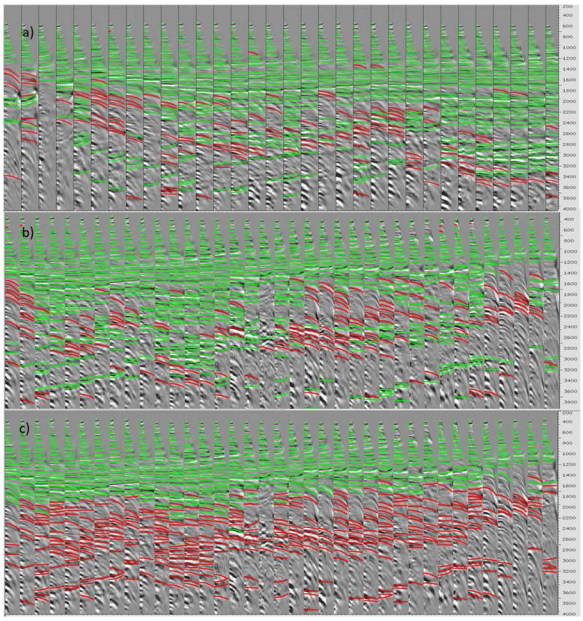

Figure 4: The training data for the larger North Sea dataset (a) and the first iteration RMO pick classifications by the Random Forest classifier (b) and the Neural Network classifier (c).

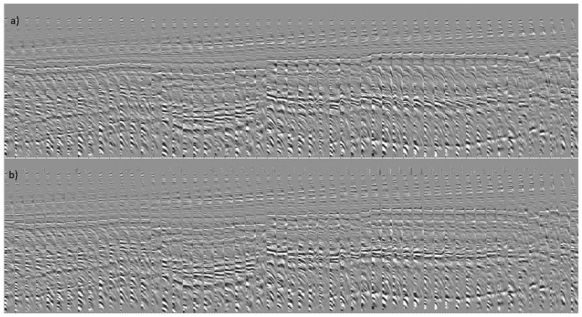

Figure 5: Migration gathers: a) initial and b) after 5 iterations of automatic velocity updating with the Random Forest classifier.

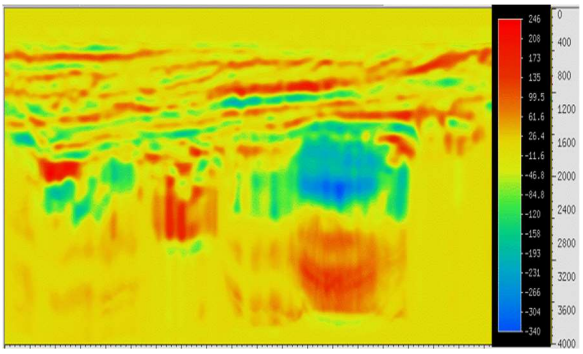

Figure 6: Velocity update after 5 iterations of automatic updating.

Conclusions and discussions

We have presented an automatic velocity model building method that combines mature model building techniques and a novel machine learning algorithm. Machine learning reduces human intervention and may enable the automation of velocity model building. The second field example shows that the Random Forest model can be trained once and applied for all iteration without the need of re-training. It means the characteristics of bad picks, for example, multiples, remain stationary.

The failure of the Neural Network classifier in the second example does not mean it is not a good machine learning technique for classifying RMO picks. It probably indicates that the hyperparameters are dataset dependent and they need to be re-tuned. On the other hand, it seems the Random Forest classifier is simpler but works reliably for identifying bad RMO picks. As Płoński (2019) suggested, when working with tubular data, it is better to start with Random Forest. If you are not satisfied with the model performance you should try to tune and train a Neural Network. There are many hyperparameters which can be tuned in a Neural Network and if you have enough knowledge and experience you can obtain very good results with Neural Networks.

Acknowledgements

The authors thank PGS for the permission to publish this paper and thank colleagues Øystein Korsmo and Tashi Tshering for providing the testing data.

REFERENCES

Araya-Polo, M., J. Jennings, A. Adler and T. Dahlke, 2018, Deep-learning tomography: The Leading Edge, **37**, 58–66, doi: 10.1190/tle37010058.1.

Krizhevsky, A., I. Sutskever and G. Hinton, 2012, ImageNet classification with deep Convolutional Neural Networks: Neural Information Processing Systems.

Øye, O. K. and E.K. Dahl, 2019, Velocity model building from raw shot gathers using machine learning: Second EAGE/PESGB Workshop on Velocities, 1–3.

Plonski, P., 2019, Random Forest vs Neural Network (classification): https://www.kdnuggets.com/2019/06/random-forest-vs-neural-network.html.

Yang, F., and J. Ma, 2019, Deep-learning inversion: A next-generation seismic velocity model building method: Geophysics, **84**, no. 4, R583–R599, doi: 10.1190/geo2018-0249.1.

Zheng, Y., Q. Zhang, A. Yusifov and Y. Shi, 2019, Applications of supervised deep learning for seismic interpretation and inversion: The Leading Edge, **38**, 526–533, doi: 10.1190/tle38070526.1.

Zhou, C., S. Brandsberg-Dahl, and J. Jiao, 2009, A continuation approach to regularize the reflection tomography with a 3D Gaussian filter: 71st Conference and Exhibition, EAGE, Expanded Abstracts, U031.

Zhou, C., J. Ramos-Martinez, S. Lin, J. Jiao, and S. Brandsberg-Dahl, 2008, True geometry tomography for velocity model building with applications to WATS seismic data: 78th Annual SEG Meeting, Expanded Abstracts, 3260–3264, doi: 10.1190/1.3064022.