Separating primaries and multiples using hyperbolic radon transform with deep learning

Harpreet Kaur, Nam Pham, and Sergey Fomel, The University of Texas at Austin.

1.

We propose a deep neural network (DNN) framework for computing hyperbolic Radon transform for separating primary reflections and multiples. The basic idea is to compute the weights associated with the inverse Hessian using the DNN for training data sets, which can then be applied to the adjoint transform of test data sets to obtain an initial model close to the true model. The output of the DNN can then be used as an input to the least-squares framework to obtain an output equivalent to the least-squares solution, but at a significantly reduced cost. Field data examples verify the effectiveness of the proposed approach.

2.

The Radon transform (Radon, 1917) is a tool that maps the overlaping events to a tranformed domain, where the events can be separated. The transform is used to focus events with linear, parabolic and hyperbolic shapes. The major difference between these transforms is that the linear and parabolic transforms are time-invariant and thus can be applied in the frequency domain using fourier transforms whereas the hyperbolic transform being time-variant has to be computed in the time domain. The parabolic Radon transform is a powerful tool but it is not perfectly suited for separating hyperbolic events. Since both primaries and multiples are hyperbolic in the time-space domain of a CMP gather, hyperbolic radon transform (also known as velocity stack or velocity transform) is applied to map them to different regions in the Radon domain corresponding to their respective velocities. After muting multiples in the Radon domain, multiple-free data can be reconstructed back to the time-space domain using inverse hyperbolic radon transform, which can then be used for further processing. Since an adjoint operation does not reconstruct data exactly, several possible solutions have been proposed to this problem, e.g., the use of weighting functions (Larner, 1979), and the use of taper windows (Blackman and Tukey, 1958). These methods reduce lateral smear of events but still have some coherent streaks remaining on the panel. Thorson and Claerbout (1985) proposed iterative least-squares inversion formulation which does not suffer from the above mentioned shortcomings. However, inversion is an expensive process and can be rather difficult to apply, especially with 3D data (Guitton, 2004). Several other methods have been proposed by different authors for separating primaries and multiples using Radon transforms (Hampson, 1986; Yilmaz, 1989; Foster and Mosher, 1992; Moore and Kostov, 2002; Hargreaves et al., 2003).

We propose to compute hyperbolic Radon transform similar to the least-squares formulation by estimating the inverse Hessian using a DNN and thus, achieve convergence at a significantly reduced cost. We further use the DNN output as an initial model to a least-squares framework to obtain the final output with fewer iterations as compared to the iterative inversion. The proposed method facilitates accurate reconstruction of data to time-space domain of a CMP gather and eliminates the need for applying a large number of iterative inversions to each CMP gather. This adaptive non-linear model makes the algorithm highly flexible and shows universitality with different data sets.

3.

Velocity stack is defined by the following equation (Thorson and Claerbout, 1985):

$$u_w(p,\tau) \equiv \int_0^\infty w(h)\, d\left(h, t=\sqrt{\tau^2+p^2h^2}\right) dh \tag{1}$$

where p is the slowness, $w(h)$ is the weighting function, $d(h,t)$ is the input to velocity stack, and (p,τ) denotes the output. In operator notation with $w(h) = 1$, it is given as:

$$u = L^T d \tag{2}$$

The offset time pair (h, t) denotes the coordinate axes of the CMP gather and the slowness pair (p, τ) denotes the coordinate axes of the corresponding velocity panel. In this transformation, there are functions belonging to two different spaces, offset space or data space, which includes all the functions defined over the (h, t) domain, and velocity space or model space, which includes all the functions defined over the (p,τ) domain. Velocity stacking performed by operator L^T using equation (2) produces lateral coherence or smearing of events in velocity panel (Thorson and Claerbout, 1985). Various choices of weighting functions are proposed by different authors to reduce the problem of smearing of events. Larner (1979) proposes a more general weighing function $w(h)$ to reduce truncation effects at far offsets. Blackman and Tukey (1958) propose a spectral method. Even with these techniques, coherent streaks still remain on the velocity stack. Thorson and Claerbout (1985) proposed least-squares formulation for this problem, which asserts that the CMP gather $d(h, t)$ is the result of some transformation on a function $u_0(p,\tau)$ in velocity space with $d(h, t)$ as :

$$d = Lu_0 + n \tag{3}$$

Least-squares approach to equation (3) estimates u to u_0 that minimizes energy in the noise term n and is given as:

$$u = (L^T L)^{-1} L^T d \tag{4}$$

The problem here is to find the inverse Hessian. We propose an algorithm using DNN to compute the inverse Hessian using equation (2) and equation (4) as inputs for training. We use a small portion of a given dataset for training and once the network learns to compute the weights associated with the inverse Hessian for training data sets, we apply learned weights to equation (2) for test data sets which are not a part of training.

The DNN output is much closer to the true model as compared to the adjoint transform, which we then use in the least-squares framework to obtain an output equivalent to the least-squares output equation (4) with fewer iterations, thus significantly reducing the computational cost.

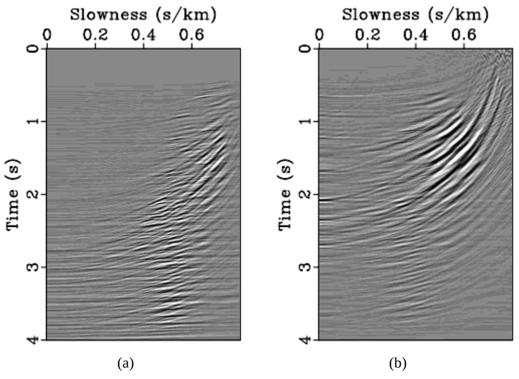

Figure 1: Velocity transform (a) Using the adjoint. (b) Using iterative least-squares.

IMPLEMENTATION OF THE PROPOSED ALGORITHM

We implement Generative Adversarial Networks (GANs) (Goodfellow et al., 2014; Zhu et al., 2017) which implicitly learn a latent, low dimensional representation of an arbitrary high dimension data. GANs are basicaly composed of two networks: a generator (G) that outputs synthesized image and a Discriminator (D) that outputs the probability between 0 and 1 corresponding to the samples being fake or real. The networks that represent the generator and the discriminator are implemented by multilayered networks consisting of convolution and/or fully connected layers (Creswell et al., 2018) where generator (G) tries to map samples from the source distribution (the adjoint velocity transform) to the target distribution (the velocity transform using iterative inversion) and discriminator (D) determines if the sample is from the actual distribution or produced by the generator. The generator and discriminator are trained in a joint framework where the generator eventually learns to approximate the underlying distribution completely, and the discriminator is left guessing randomly. Further, the network uses a loss function to reduce the space of possible mapping functions (Kaur et al., 2019).

4.

For training and validation of the algorithm we use the Viking Graben data set consisting of common mid point (CMP) gathers. We divide this data set into training data consisting of five CMP gathers and test data consisting of 95 CMP gathers. For training we divide the CMP gathers into patches of 160*160 samples. We train the network in the model domain using the velocity transform with the adjoint operator $(L^T d)$ and the velocity transform with the least-squares inverse operator $(L^T\ L)^{-1}\ L^T\ d$ such that the network learns the relationship between the two, i.e., the weights associated with the inverse Hessian $(L^T L)^{-1}$ operator. For creating the training data we start with a CMP gather, perform a velocity transform using the adjoint operator as shown in Figure 1a, as well as a velocity transform using a least-squares inversion with 20 conjugate gradient iterations as shown Figure 1b. Using these two inputs, DNN learns to compute the inverse Hessian, which we then apply to the adjoint velocity transform to obtain a least-squares velocity transform. After training, we test the efficiency of the algorithm with the CMP gathers unseen during the training process which comprise the test data set. Figure 2b shows one of the test CMP gathers with the velocity transform using the adjoint operator applied to the CMP gather shown in Figure 2a. Coherent streaks in Figure 2b produced by L^T are artifacts resulting primarily from the truncation of events at the highest offset trace on the CMP gather. The network applies the inverse Hessian operator learned during training to the adjoint velocity transform shown in Figure 2b to obtain results as shown in Figure 2c which are close to the least-squares solution. The DNN output shown in Figure 2c needs further improvement, therefore we use this as an initial model to an iterative least-squares framework and call the output the deep neural network least-squares solution (DNNLS). The DNNLS output is shown in Figure 2d which is similar to the least-squares inversion solution as shown in Figure 2e. Data reconstructed using the adjoint transform (filtered back projection), the inverse transform using DNNLS, and the inverse transform using iterative inversion are shown in Figure 2f, 2g, and 2h, respectively. The CPU time for the adjoint operation is 5.85 s and 235.64 s for the iterative inversion using 20 iterations. DNN test runs in only 0.31 s of CPU time for each cmp gather. To make it faster, we can further use a fast butterfly algorithm for the adjoint part (Hu et al., 2013) and use the proposed algorithm to achieve results similar to the iterative inversion. The fast butterfly algorithm takes 3.87 s for the adjoint part compared to 5.85 s taken by the conventional algorithm. Figure 3a and 3b show the separated signal (primary reflections) using DNNLS and iterative inversion, respectively with velocity analysis using the semblance scan with DNNLS and the iterative inversion shown in Figure 3c and 3d. Separated noise (multiple reflections) using DNNLS and the iterative inversion are shown in Figure 3e and 3f with velocity analysis using semblance scan for multiples with DNNLS and iterative inversion shown in Figure 3g and 3h. Normal moveout- corrected signal using the DNNLS picked primary velocity is shown in Figure 3i.

5.

We have introduced a novel way of estimating hyperbolic radon transforms using a deep neural network to achieve faster convergence at a significantly reduced cost. A field example demonstrates that the proposed algorithm significantly reduces the lateral smearing of events in the model space and is capable of reconstructing data space from model space with accuracy similar to the least-squares inversion. The proposed algorithm is cost effective and efficient, hence, it can be used in the application of the separation of primaries and multiples.

6.

Blackman, R. B., and J. W. Tukey, 1958, The measurement of power spectra: Dover Publications Inc.
Creswell, A., T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, 2018, Generative adversarial networks: An overview: IEEE Signal Processing Magazine, 35, 53–65, doi: https://doi.org/10.1109/MSP.2017.2765202.
Foster, D. J., and C. C. Mosher, 1992, Suppression of multiple reflections using the Radon transform: Geophysics, 57, 386–395, doi: https://doi.org/10.1190/1.1443253.
Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, 2014, Generative adversarial nets: Advances in Neural Information Processing Systems, 2672–2680.
Guitton, A., 2004, Amplitude and kinematic corrections of migrated images for nonunitary imaging operators: Geophysics, 69, 1017–1024, doi: https://doi.org/10.1190/1.1778244.
Hampson, D., 1986, Inverse velocity stacking for multiple elimination: 56th Annual International Meeting, SEG, Expanded Abstracts, 422–424, doi: https://doi.org/10.1190/1.1893060.
Hargreaves, N., B. verWest, R. Wombell, and D. Trad, 2003, Multiple attenuation using an apex-shifted Radon transform: 73rd Annual International Meeting, SEG, Expanded Abstracts, 1929–1932, doi: https://doi.org/10.4043/16944-MS.
Hu, J., S. Fomel, L. Demanet, and L. Ying, 2013, A fast butterfly algorithm for generalized Radon transforms: Geophysics, 78, no. 4, U41–U51, doi: https://doi.org/10.1190/geo2012-0240.1.
Kaur, H., N. Pham, and S. Fomel, 2019, Seismic data interpolation using CycleGAN: 89th Annual International Meeting, SEG, Expanded Abstracts, 2202–2206, doi: https://doi.org/10.1190/segam2019-3207424.1.
Larner, K., 1979, Optimum-weight CDP stacking: Western Geophysical Co.: unpublished notes. Moore, I., and C. Kostov, 2002, Stable, efficient, high-resolution Radon transforms: 64th Conference and Exhibition, EAGE, Extended Abstracts, cp–5.
Radon, J., 1917, On the determination of functions from their integrals along certain manifolds: Ber. Verh, Sachs Akad Wiss., 69, 262–277.
Thorson, J. R., and J. F. Claerbout, 1985, Velocity-stack and slant-stack stochastic inversion: Geophysics, 50, 2727–2741, doi: https://doi.org/10.1190/1.1441893.
Yilmaz, Ö., 1989, Velocity-stack processing1: Geophysical Prospecting, 37, 357–382, doi: https://doi.org/10.1111/j.1365-2478.1989.tb02211.x.
Zhu, J.-Y., T. Park, P. Isola, and A. A. Efros, 2017, Unpaired image-to-image translation using cycle-consistent adversarial networks: arXiv preprint.