How much information is in a seismogram?
Autoencoder networks for seismic data compression

Andrew Valentine & Jeannot Trampert • Department of Earth Sciences, Universiteit Utrecht
andrew@geo.uu.nl

Universiteit Utrecht

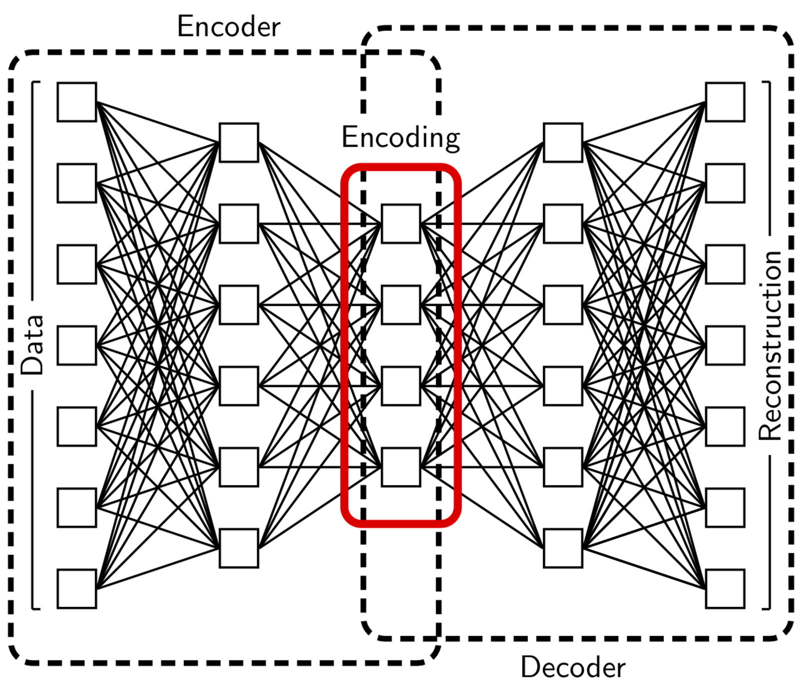
Overview

Seismograms tend to be quite distinctive; an experienced seismologist can easily distinguish between seismic data and many other time series. What does this mean? In effect, an N-point time series may be regarded as a single point in N-dimensional space. However, N-point seismograms occupy only a subset of this space; in effect, they exist in a lower-dimensional space. What is the dimension of this space, and how can we explore it? How does it vary with different classes of seismic data?

Hinton & Salakhutdinov (2006) showed that a class of neural networks known as 'autoencoders' can be used to find lower-dimensional structure within a dataset, by attempting to construct a lossless representation of each datum in a lower-dimensional space. We consider how this might be applied to seismic data, and what possible applications are revealed.

Autoencoder networks

An 'autoencoder' is a network trained to output a faithful representation of its inputs. Its architecture is such that there are fewer nodes in hidden layers than in the input/output layers. The values of nodes in a hidden layer can then be taken as an encoded form of the inputs, and the autoencoder may be regarded as an encoder/decoder pair.



Autoencoders are described by specifying the number of nodes per layer; the above therefore depicts a 7-6-4-6-7 autoencoder. We use logistic neurons, which implement

$$f(x) = f_0 + \frac{f_1 - f_0}{1 + \exp(-x)},$$

for constants f_0, f_1. We denote the values of the n-th layer of *nodes* by $\mathbf{x}^{(n)}$. Associated with each neuron are weights corresponding to each input, $\mathbf{W}$, a bias, b, and a sensitivity, a. For the i-th element of $\mathbf{x}^{(n)}$, we therefore have

$$x_i^{(n)} = f\left(a_i^{(n)} b_i^{(n)} + a_i^{(n)} \sum_j W_{ij}^{(n)} x_j^{(n-1)}\right).$$

We define a measure of the difference between L network inputs, $\mathbf{x}^{(0)}$, and outputs, $\mathbf{x}^{(N)}$, across a dataset of M examples

$$E = \frac{1}{2}\sum_i^L \sum_j^M \left(x_{ij}^{(N)} - x_{ij}^{(0)}\right)^2,$$

and we adjust W_{ij}, a_i and b_i to reduce this error. This may be achieved by updates according to

$$b_i^{(n)} \to b_i^{(n)} - \frac{\eta}{M}\sum_j \Delta_{ij}^{(n)} u_{ij}^{(n)} a_i^{(n)},$$

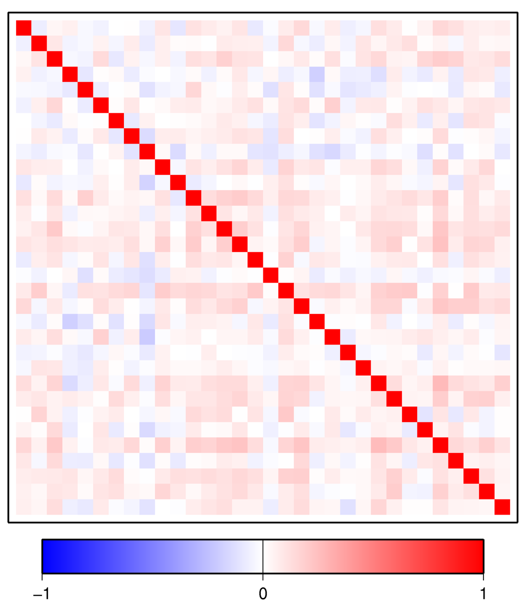
$$W_{ij}^{(n)} \to W_{ij}^{(n)} - \frac{\eta}{M}\sum_k \Delta_{ik}^{(n)} a_i^{(n)} u_{ik}^{(n)} x_{jk}^{(n-1)},$$

$$a_i^{(n)} \to a_i^{(n)} - \frac{\eta}{M}\sum_j \Delta_{ij}^{(n)} u_{ij}^{(n)} \left(b_i^{(n)} + \sum_k W_{ik}^{(n)} x_{kj}^{(n-1)}\right).$$

Here, η is a *learning rate parameter*, controlling the amount of information the network assimilates at each step. Repeated application of these rules is necessary, owing to the inherent non-linearity of the system.

Demonstration

- Construct and train a 512-256-128-64-32-64-128-256-512 autoencoder.
- Training dataset: 880 good-quality 512-point seismograms chosen at random from magnitude 6+ events in 2000; sampled at 16-second intervals, filtered to contain frequencies below 7.4 mHz.
- Monitoring dataset: 276 good-quality 512-point seismograms, chosen similarly to training dataset. Not provided to network during training.
- 500 CRBM training iterations; 500 training iterations using assembled autoencoder.

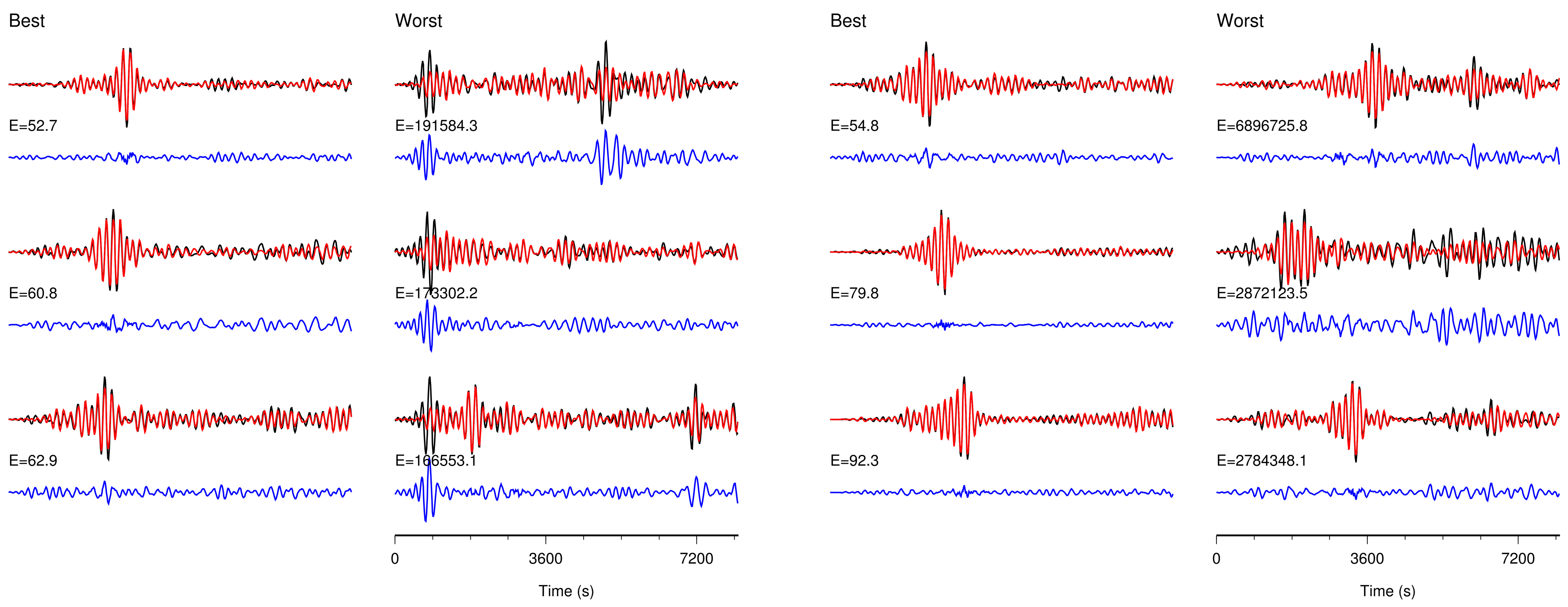


Left: A 'basis'? 32 waveforms $\mathbf{b}_i$ generated by decoding the unit vectors $(1, 0, \ldots, 0)$, $(0, 1, \ldots, 0)$ etc. Figure shows 'orthogonality' matrix

$$M_{ij} = \frac{\mathbf{b}_i \cdot \mathbf{b}_j}{\mathbf{b}_i \cdot \mathbf{b}_j}$$

Note, however, that our decomposition is non-linear.

We take 512-point waveforms (black), encode them in a 32-element representation, and then decode (red). We find a good agreement (blue). Shown are the best and worst three traces in the training set (left) and monitoring set (right).



Pre-training the autoencoder

Autoencoder training from scratch is slow, and for complex datasets non-linearity may prevent satisfactory progress. Hinton & Salakhutdinov (2006) demonstrate that this can be circumvented via a layer-by-layer pre-training stage. For this, we make use of *Continuous Restricted Boltzmann Machines* (CRBMs) – see Chen & Murray (2003). These are two-layer networks, with a stochastic relationship between layers. The visible nodes, $\mathbf{x}^v$, are used to update the hidden nodes ,$\mathbf{x}^h$, according to

$$x_i^{\mathrm{h}} = f\left(a_i^{\mathrm{h}}\left(b_i^{\mathrm{h}} + \sum_{j=1}^N w_{ij} x_j^{\mathrm{v}} + \mathcal{G}(0,\sigma)\right)\right),$$

with $\mathcal{G}(\mu, \sigma)$ representing a random sample from a Gaussian distribution of mean μ and standard deviation σ. Similarly, the hidden nodes may be used to update the visible nodes:

$$x_j^{\mathrm{v}} = f\left(a_j^{\mathrm{v}}\left(b_j^{\mathrm{v}} + \sum_{i=1}^N w_{ij} x_i^{\mathrm{h}} + \mathcal{G}(0,\sigma)\right)\right).$$

The visible-to-hidden and hidden-to-visible connections share (transposed) weight matrices, but have independent biases and sensitivities, and the CRBM training rules seek to find and enhance correlations between visible and hidden nodes (Chen & Murray, 2003)

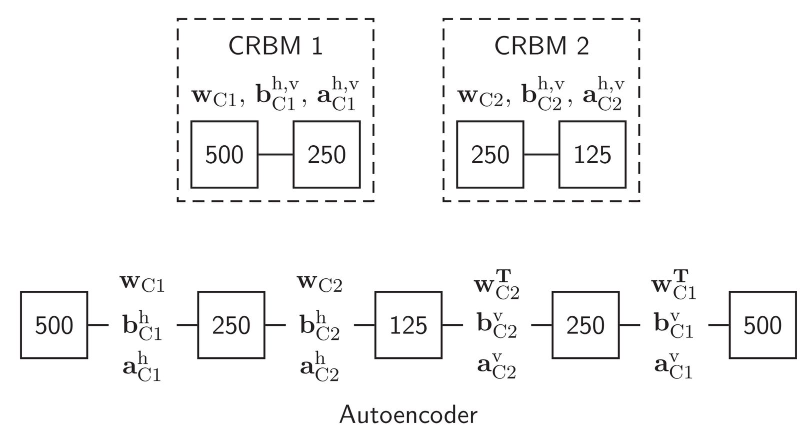
$$b_i^{\mathrm{h,v}} \to b_i^{\mathrm{h,v}} + \eta\left[\left\langle x_i^{\mathrm{h,v}}\right\rangle - \left\langle \hat{x}_i^{\mathrm{h,v}}\right\rangle\right],$$

$$w_{ij} \to w_{ij} + \eta\left[\left\langle x_i^{\mathrm{h}} x_j^{\mathrm{v}}\right\rangle - \left\langle \hat{x}_i^{\mathrm{h}} \hat{x}_j^{\mathrm{v}}\right\rangle\right],$$

$$a_i^{\mathrm{h,v}} \to a_i^{\mathrm{h,v}} + \frac{\eta}{\left(a_i^{\mathrm{h,v}}\right)^2}\left[\left\langle \left(x_i^{\mathrm{h,v}}\right)^2\right\rangle - \left\langle \left(\hat{x}_i^{\mathrm{h,v}}\right)^2\right\rangle\right].$$

where angled brackets $\langle\chi\rangle$ denote the average value of χ across all samples in the training set, and 'hats' denote values when the CRBM is encoding its own outputs. Again, η acts as a learning rate parameter.

Suppose we wish to construct a 500-250-125-250-500 autoencoder. We begin by creating a CRBM with 500 visible and 250 hidden nodes. After training for a number of iterations, we use this to convert our dataset of 500-element vectors into 250-element vectors. This reduced dataset is then used to train a CRBM with 250 visible and 125 hidden nodes. This may be used to assemble a pre-trained autoencoder, as shown below.



Applications

There are a number of potential applications of the autoencoder method, and directions for further investigation:

- Quality control – good-quality traces can be recovered accurately after encoding, noisy traces cannot. Can this be used to identify high-quality traces in seismic databases?
- Noise removal – if a trace containing moderate noise is encoded and recovered, is the resulting trace 'cleaner' than the original?
- Sorting and searching of databases – can we relate waveform characteristics to particular aspects of their encoded representations?
- Non-linear tomography – tomographic methods based on neural networks are attractive, but computationally challenging. Reducing the dimension of the data-space is therefore extremely beneficial.
- Can computation be carried out in the encoding domain?

Acknowledgements

QUANTITATIVE ESTIMATION OF EARTH'S SEISMIC SOURCES AND STRUCTURE — QUEST

MARIE CURIE ACTIONS

References

Chen, H. & Murray, A., 2003. Continuous restricted Boltzmann machine with an implementable training algorithm, *Vision, Image and Signal Processing, IEE Proceedings*, 150, 153–158.
Hinton, G. & Salakhutdinov, R., 2006. Reducing the Dimensionality of Data with Neural Networks, *Science*, 313, 504–507.
Valentine, A. & Trampert, J., *in prep.* Compression, quality assessment and searching of waveforms: Data-space reduction via autoencoder networks.