An Exploration of Auto Encoders on MNIST

Jonathan E. Mathews
Department of Electrical and Computer Engineering
and
Department of Mechanical Engineering
South Dakota School of Mines and Technology
Rapid City, SD 57701
jonathan.mathews@mines.sdsmt.edu

Abstract

Auto-encoders are an important field in unsupervised learning today. They are used for data compression, manifold learning, and de-noising. This paper will examine the significance of the auto-encoder for data analysis in engineering and science. Three different classes of auto-encoder will then be tested on the MNIST data set to compare their effectiveness for data compression and manifold learning.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

1 Introduction

Supervised machine learning has been used to solve complex classification problems but it requires large amounts of preclassified data for training. Unsupervised machine learning attempts to solve this problem by extracting order from data without a pre-determined solution to compare with. Auto encoders are a specific class of unsupervised network that do this by transforming the data into a latent space with desirable properties and then transforming the latent representation back into a form resembling the original input. The loss function usually involves the accuracy of the reconstruction combined with a parameter representing the conformance of the latent space to the desired form.

One of the common goals of an auto-encoder is to find a lower-dimensional representation of the data that, with the appropriate decoder, can be used to reconstruct the original high-dimensional input with minimal reconstruction loss. This is contrasted with a denoising auto-encoder that may have a higher-dimensional latent space than the input, but whose purpose is to reconstruct the original data given a noisy version of the original input. Reducing the dimensionality of the latent-space compresses the data and can aid in classification by allowing the classification to train on a lower-dimensional input that still represents a comparable amount of information to the original input.

This paper will explore the properties of three different compressive auto-encoders and their facility for data compression. First, this paper will provide a discussion of a linear auto-encoder and its cognates in the machine learning world. Next, the structure will be extended to use a variety of non-linear activation functions with the goal of improving reconstruction performance under the assumption of a non-linear manifold. Finally, the variational auto-encoder will be explored and it's capability to extract the data manifold will be contrasted with the other auto-encoder methods.

2 The significance of manifold learning

The crux of solving problems in engineering and science is often data visualization. The real world is full of phenomena that distort and obscure the desired trends. Finding an appropriate transformation that moves the data from the realm of seeming chaos to the realm of interpretable results is therefore key. In the thermal sciences, this is often done by non-dimensionalizing the data to extend applicability to untested cases. In control theory, the LaPlace and Fourier transforms can be used to show the frequency-dependence of the system. Even something so simple as using a logarithmic plot instead of a linear plot is an example of data transformation for visualization.

The fundamental concept behind manifold learning is that the physical processes that give rise to the data are relatively few in number and typically exist in a continuous space. A simple example would be using a camera to record the trajectory of a soccer-ball after it was kicked. The problem could be somewhat accurately simulated using a small number of parameters such as the impulse imparted to the ball by the kick, the point of contact between foot and ball, the presence of wind or friction of the grass, the shape of the ball, and the mass of the ball. This is only six parameters that could vary continuously in the space of all experiments.

If the measurement system is a 720 x1080 pixel rgb camera at 30-fps, and each experiment takes 5 seconds, then the raw measurement data contains 349,920,000 individual pixel values between 0 and 255. Obviously, there is a huge amount of unnecessary data collected by the system. Manifold learning says that if we represent each experiment as a single point in 349,920,000-dimensional space, all of the experiments should lie close to some 6-dimensional manifold or surface where traversing the manifold is equivalent to traversing the original six physical parameters. It concerns itself with finding an appropriate transform that can map the results of a new experiment to a low-dimensional space where each dimension has a direct physical corollary.

This concept coincides with the function of an auto-encoder. The encoding portion of an auto-encoder takes a high-dimensional input and reduces it to a low dimensional latent-space. This would correspond to taking the video of the soccer ball and extracting from it the kicking parameters. The decoding portion takes that latent-space and maps it back to the high-dimensional space. In the soccer example, if the latent-space truly represents the physical variables in the system, then the decoder could also be thought of as a simulator that could reproduce an accurate video of a soccer-ball being kicked based on the desired physical parameters.

An auto-encoder can therefore be thought of as a deep network that attempts to learn the relationship between data and the processes that generated the data. It guesses the appropriate encoding and decoding relationships and then tries to extract the latent parameters and simulate the result. It then compares the simulation with the original input and iterates until it finds a solution that minimizes the difference between the measured input and the simulated result. In this way, the complex relationships between process and data can be determined and a useful visualization developed.

This paper will explore auto-encoders using the MNIST dataset which consists of 28x28-pixel images of hand-drawn numeric digits from 0 to 9. The physical process used to create those images was a person moving a pen across paper. We would therefore expect the manifold extracted to have some correspondence to the various strokes of a pen with some additional parameters to account for the variation within each stroke. There are perhaps 10 to 20 distinct strokes that are used to make each character along with a few parameters each to represent degree of curvature, width of each stroke, etc. The hypothesis is therefore that an appropriate latent-space will have a dimensionality of $10^1 < N < 10^2$ while the input has 784 dimensions.

3 Linear auto-encoder

A linear auto-encoder resolves the input onto a low-dimensional plane. A cognate in the machine learning world is Principal Component Analysis (PCA). In PCA, the data is resolved onto an n-dimensional subspace with the goal of maximizing the variance of the data within that subspace [1]. This is accomplished by calculating the eigenvectors of the data and choosing the n eigenvectors that have the highest eigenvalues as the basis for the latent space. This process can be approximated by an auto-encoder by using strictly linear activations. The resulting latent space should span a similar subspace to the first n eigenvectors, but it will not guarantee that the basis vectors are ranked by significance, or that they are orthonormal as PCA does.

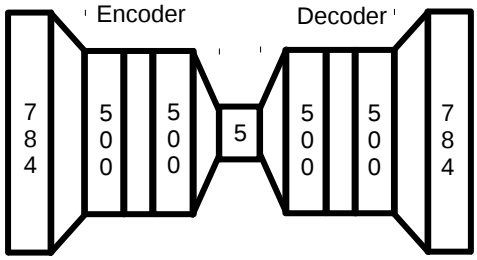

Figure 1: Auto-encoder structure used for the experiments in this report. The numbers indicate the dimensionality of each layer.

To test the linear network's capability as an auto-encoder on MNIST, a network was defined as shown in Figure 1 that used two hidden layers with 500 nodes in both the encoder and decoder. The latent dimension was set to 5 based on some preliminary testing. The loss function used was the cross-entropy between the input and reconstructed images defined as:

$$L = -\sum \left[x_{in} \cdot log(x_{out}) + (1 - x_{in}) \cdot log(1 - x_{out}) \right] \tag{1}$$

Figure 2 shows the results of the linear encode-decode process at three points during a 50,000 step training run using 100-image batches. It is clear that the network is learning because a few samples in the final reconstruction can be correctly identified by eye. The reconstruction is quite bad though. The network also requires a long time to converge to an identifiable reconstruction when compared with other methods.

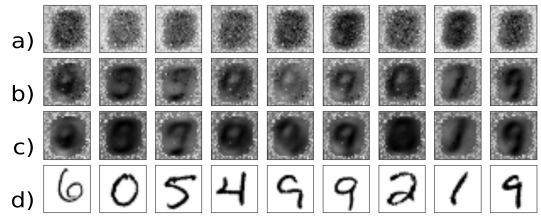

Figure 2: Results of linear encoding and decoding after **a)** 10,000, **b)** 30,000, **c)** 50,000 training steps compared with **d)** the input images

4 Non-linear auto-encoders

Although a linear auto-encoder may work well for a subset of problems where the data lies in a linear subspace of the measurement space, it does not work well on MNIST. This is logical because the process of writing a digit is intuitively non-linear. For this reason, three different non-linear transforms were applied to each of the 500-node hidden layers and their results were compared with those of the linear approach. The activation functions were:

- Sigmoid
- Rectified Linear Units (ReLU)
- Exponential Linear Units (ELU)

The loss function was left identical to that used for the linear approach.

Figure 3 shows the results after training each of the non-linear models for 50,000 steps with a batch size of 100 samples like the linear model. All of the non-linear activations yielded much better results than the linear model and are virtually indistinguishable from each other.

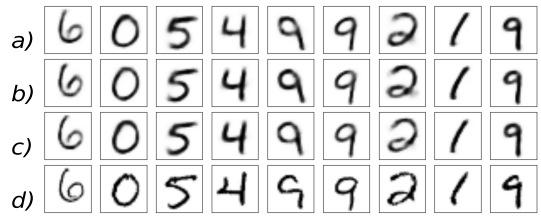

Figure 3: Results of nonlinear encoding and decoding with a **a)** sigmoid, **b)** ReLU, and **c)** ELU activation functions compared with **d)** the input images

5 Variational auto-encoder

The variational auto-encoder (VAE) approaches the data representation problem from a probabalistic standpoint. From Kingma et al.:

> We assume that the data are generated by some random process, involving an unobserved continuous random variable $\mathbf{z}$. The process consists of two steps: (1) a value $\mathbf{z}^{(i)}$ is generated from some prior distribution $p_\theta(\mathbf{z})$; (2) a value $\mathbf{x}^{(i)}$ is generated from some conditional distribution $p_\theta(\mathbf{x}|\mathbf{z})$.[2]

In this model, θ represents a set of parameters that define the probability distribution. To return to the soccer illustration, the variational model assumes that the set of all possible kick parameters falls within some probability distribution, $p_\theta(\mathbf{z})$, while a single trajectory measurement for a given set of kick parameters falls within a joint distribution, $p_\theta(\mathbf{x}, \mathbf{z})$. According to Bayes' rule, $p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{z}|\mathbf{x})p_\theta(\mathbf{x}) = p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})$. Thus we have that $p_\theta(\mathbf{x}|\mathbf{z})$ represents how likely it is to get a certain trajectory video given specific parameters for the kick. The expectation of this distribution is the most

likely trajectory for any given kick. We will refer to this as a probabalistic decoder. Also, $p_\theta(\mathbf{z}|\mathbf{x})$ represents the likelihood that the kick had certain parameters given that the trajectory is as measured. The expectation of this distribution would be the most likely kick parameters given any trajectory video. We refer to this as a probabalistic encoder [2].

According to Kingma, the true probabalistic encoder model is often intractable, so instead it will be approximated by a recognition model, $q_\phi(\mathbf{z}|\mathbf{x})$ that we define to be Gaussian[2]. This means that a deterministic function can be derived that maps any value of **x** to the mean and standard deviation of **z**. In essence, this says that a certain trajectory most likely came from the mean that the mapping produces, but it could have come from any kick with similar parameters. Because it's Gaussian, the farther we get from the mean, the less likely it is that the observed trajectory was produced by that kick.

The resulting deep network structure is shown in Figure 4. There are two main differences between the simple non-linear auto-encoder and the variational auto encoder. The first is that the latent space is not passed directly through from encoder to decoder. Instead, a sample is taken from the spherical gaussian distribution defined by the encoder and passed through to the decoder. The second is that the loss function includes a term to account for the KL-divergence between $p_\theta(\mathbf{z}|\mathbf{x})$ and $q_\phi(\mathbf{z}|\mathbf{x})$. This term essentially tries to ensure that the approximated prior distribution is as close as possible to the actual prior distribution.

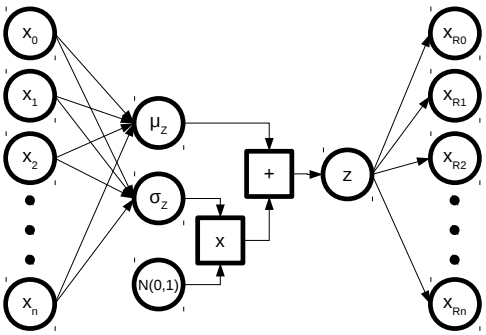

Figure 4: The basic structure of the variational auto-encoder. The networks in the encoder and decoder each contain 2 densely connected layers of 500 nodes each with ELU activation functions.

Figure 5 shows the reconstruction produced by the VAE. It provides comparable results to the simple non-linear auto-encoders, but has the added advantage that continuous variation of the latent-space parameters will result in continuous variation of the output within the domain of hand-drawn digits. This implies that the VAE is actually quite close to the manifold representing the 5^{th} dimensional surface in 784-dimensional space that corresponds to hand-drawn digits.

Figure 5: The reconstruction produced by **a)** the VAE, compared with **b)** the input.

6 Comparison of auto-encoders

The linear auto-encoder demonstrated worse reconstruction loss than any of the other encoders tested. It also took much longer to converge. Figure 6 shows the reduction in reconstruction loss during training, but it is clear that the linear network is orders of magnitude worse than any of the other networks. Figure 7 compares the training of each of the non-linear networks and the VAE. As far as reconstruction loss goes, they are very similar while converging. After the non-linear networks

converge, they are truly indistinguishable, with each network being slightly better than the others some portion of the time.

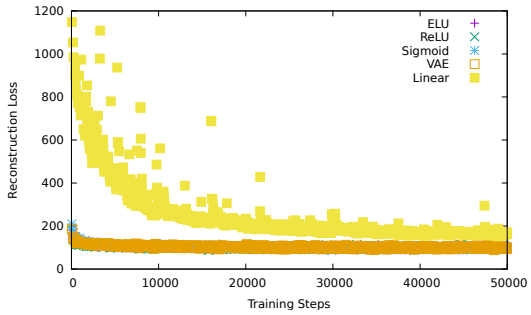

Figure 6: Reconstruction loss vs training step for all of the networks tested.

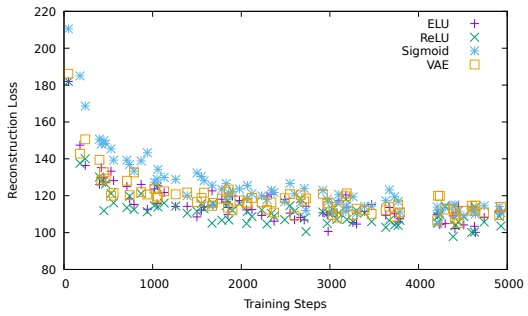

Figure 7: Reconstruction loss vs training step for all but the linear network.

7 Conclusions

These experiments showed that a small network can be used to compress data down to a much lower dimensional space than the input while still retaining enough information to reconstruct a recognizable version of the input. Non-linear activation functions were found to be necessary to obtain good results on the MNIST dataset, but the specific type of activation was found to be insignificant in terms of reconstruction performance. The variational auto-encoder was found to yield similar reconstruction performance to the other non-linear auto-encoders and has the theoretical benefit of finding a representation that more closely fits the real manifold that the data was generated from.

Acknowledgments

I would like to thank Dr. Hoover for inspiring this particular avenue of exploration and for teaching a valuable segment on manifold learning concepts. I hope to continue exploring this field in the future.

I would also like to thank Miriam from the Fast Forward Labs blog for her excellent coverage of Variational Auto Encoders on MNIST. Her `Dense` class was invaluable for producing networks in tensorflow that could be fed and accessed from multiple points. Her implementation of the Variational Auto Encoder helped me avoid a few training pitfalls due to vanishing or exploding gradients and lack of L-2 regularization also.

References

[1] Randy Hoover. *An Introduction to Manifold Learning*. Mar. 2018.

[2] D. P Kingma and M. Welling. “Auto-Encoding Variational Bayes”. In: *ArXiv e-prints* (Dec. 2013). arXiv: `1312.6114 [stat.ML]`.