Litterature review - Autoencoders

Département de mathématiques et de statistique - Université Laval

Stéphane Caron

22 September, 2019

Contents

1 Introduction to autoencoders

An autoencoder is a neural network technique that aims to learn an input and output it back as its output (Goodfellow, Bengio, and Courville 2016). To achieve this objective, the autoencoder has 2 components: an **encoder** and a **decoder**. The encoder receive an input x and and converts it to a hidden representation z. The decoder receive a representation z and decode it back to retrieve as much as possible the input x. Historically, autoencoders were known as a dimensionnalty reduction method, but it has now more applications by learning latent variables useful in generative modelling.

1.1 Architecture

As explained in the introduction, autoencoders are divided in two parts: the encoder and the decoder. The encoder compress the information into a given representation and the decoder decompress it back to its original format (see figure 1).

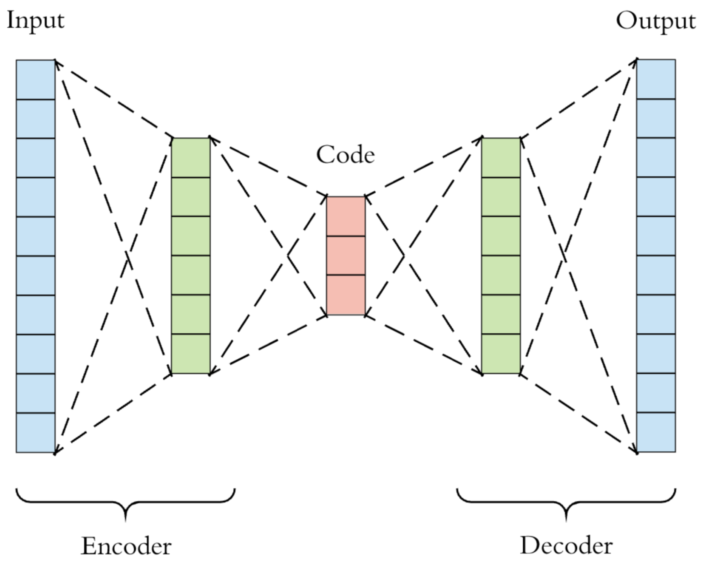

Figure 1: Autoencoders structure example.

The hidden representation (or latent representation) is often smaller because of the structure of the layers between the input and that representation. Autoencoders that have a smaller hidden representation than its inputs are called *undercomplete*. In those cases, the constraint on z force the network to learn the more important features. It could applied to feedforward networks or a convolutional networks. We are typically interested in that hidden representation, in comparison to the output of the decoder who may sound useless.

1.2 Optimization

Autoencoders intuition is to build back the input x by passing through the 2 components (encoder and decoder). As such, this kind of model do not need any target, so we say it is an unsupervised method. The training of the parameters is mainly done by minimizing the reconstruction error. The loss is given by:

$$L(x, p_\phi(q_\theta(x)))$$

where $q(x,\theta)$ is the encoder and $p(z,\phi)$ is the decoder function. The loss generally includes another component $\Omega(x)$ that acts as a regularizer. The regularization is often essential to prevent the network to "copy/paste" examples of the training set. Those situations can occur even more when the hidden representation is equal or greater than the inputs (**overcomplete**). The minimization of this loss function is done by gradient descent. For instance, the encoder parameters are gradually updated by:

$$\theta \leftarrow \theta - \epsilon * \frac{\partial L}{\partial \theta}$$

where ϵ is the learning weight.

2 Types of autoencoders

In the first sections, I introduced the general idea behind the autoencoders. However, there are different kinds of autoencoders and each has strengths in different areas. They differ in their architecture and optimization process, but at the end they all consist in unsupervised methods that learn latent structure of the data.

2.1 Sparse autoencoders

One way of learning useful representations using autoencoder is to impose a constraint of **sparsity** in the hidden representation (Ranzato et al. 2007). A sparse autoencoder is basically an autoencoder who has a sparsity penalty in the loss criterion:

$$L(x, p_\phi(q_\theta(x))) + \Omega(z)$$

where z is the hidden representation (or the output of the encoder). That sparsity penalty will pull some neurons output values towards zero in the hidden layer, making them "inactive" (Ng 2011). That regularization will prevent overfitting and prevent the network to learn by heart every input.

Sparse autoencoders are generally used to learn features for another task (e.g classification). In that sense, it can accomplish the feature engineering of a task by learning useful features.

2.2 Denoising autoencoders

Another kind of autoencoder that has been widely used are the denoising autocoders (Vincent et al. 2008). These autoencoders are really close to "vanilla autoencoders" except they receive a corrupted input and afterwards are trained to produce a uncorrupted output.

The denoising autoencoders will then minimize:

$$L(x, p_\phi(q_\theta(\tilde{x})))$$

where $\tilde{x}$ is a copy of x that has been corrupted.

There are different ways of corrupting the input. One way could be to choose a given proportion ν of inputs to randomly "destruct" by assigning a 0-value while keeping the other untouched. That approach is called *masking noise*. We could also corrupt the data by adding *Gaussian noise* or setting randomly a proportion of the inputs to maximum or minimum value (*salt-and-pepper noise)*. The intuition behind this approach is again to prevent the algorithm from learning the useless identity function.

2.3 Contractive autoencoders

The contractive autoencoders (Rifai et al. 2011) are often associated with the denoising autoencoders. The loss criterion includes a regularizer penalty in the form of:

$$\Omega = \lambda \left|\left| \frac{\partial q_\theta(x)}{\partial x} \right|\right|_F^2$$

Denoising autoencoders are build to make the reconstruction function resist small but finite-size perturbations of the inputs, while contractive autoencoders make the feature extraction function resist small changes in the inputs.

2.4 Variational autoencoders

The variational autoencoders (Kingma and Welling 2013) have a slightly different approach than other kind of autoencoders and are particularly useful in generative modelling and reinforcement learning. Again, these autoencoders have a loss criterion in the form of:

$$L = \text{reconstruction error} + \text{regularizer}$$

However, instead of encoding a hidden representation of size n, it outputs two vectors of size n: a vector of means $\boldsymbol{\mu}$ and a vector of standard deviation $\boldsymbol{\sigma}$. Those vectors allows to sample from Gaussian distributions, which makes the variational autoencoders unique property of having a latent space that is **continuous** (Shafkat 2018). This is the main difference with *vanilla autoencoders*. That property is actually very useful in generative modelling. The figure 2 illustrates the basic structure of a variational autoencoder.

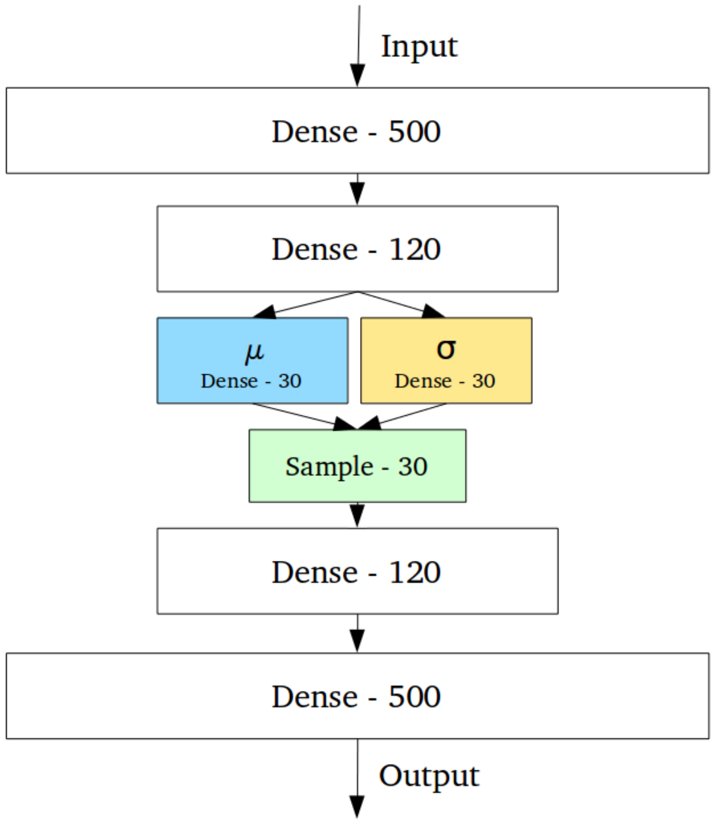

Figure 2: Variational autoencoder structure. Instead of learning a representation directly, it learns the parameters of Gaussian distributions and the hidden representation is given by sampling from those distributions.

The explicit variational autoencoder loss criterion is given by :

$$L(x, p_\phi(q_\theta(x))) + D_{KL}(q_\theta(z|x)||p_\phi(x|z))$$

where D_{KL} is a regularizer called the Kullback-Leibler divergence. His goal is to ensure that the two distributions p and q are somewhat similar.

3 Known applications of autoencoders

3.1 Dimensionnality reduction

One of the first application for which the autoencoders were used was to **reduce the dimensionnality**. For example, (Hinton and Salakhutdinov 2006) showed that a deep autoencoder could learn to represent data in a smaller representation better than the widely used PCA method (see figure 3).

Fig. 3. (A) The two-dimensional codes for 500 digits of each class produced by taking the first two principal components of all 60,000 training images. **(B)** The two-dimensional codes found by a 784-1000-500-250-2 autoencoder. For an alternative visualization, see (*8*).

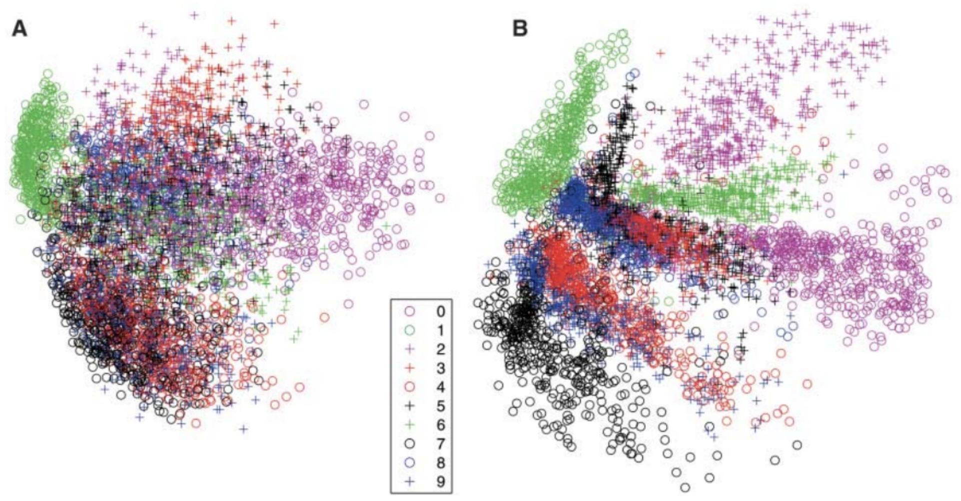

Figure 3: Comparison between the learned representation of a PCA and a deep autoencoder.

3.2 Information retrieval

Another application of autoencoders is to related with **information retrieval**. This is the task of finding an entry similar to another entry in a database. It can be done by dimensionnality reduction, and even binarization of an input, which is called semantic hashing. This has been applied to both text (Salakhutdinov and Hinton 2009) and images (Weiss, Torralba, and Fergus 2008).

3.3 Anomaly detection

A task that is often related with unsupervised methods is the idea of finding outliers or anomaly in a dataset. In fact, that's normally an information we don't have and that we can't train on it. Thus, we generally need unsupervised way of discovering those anomalies.

For instance, (Zhou and Paffenroth 2017) used denoising autoencoders to build an unsupervised anomaly detection algorithm. They applied their methodology, that includes a anomaly regularizing penalty, on MNIST dataset to find anomalies in hand written digits.

Another interesting example of anomaly detection using autoencoders was made by (Zong et al. 2018). They used autoencoders to build a small representation and use that representation as well as the reconstruction error to train a Gaussian Mixture Model (GMM). The figure 4 shows the overview of the compression (deep autoencoder) and estimation (GMM) tasks.

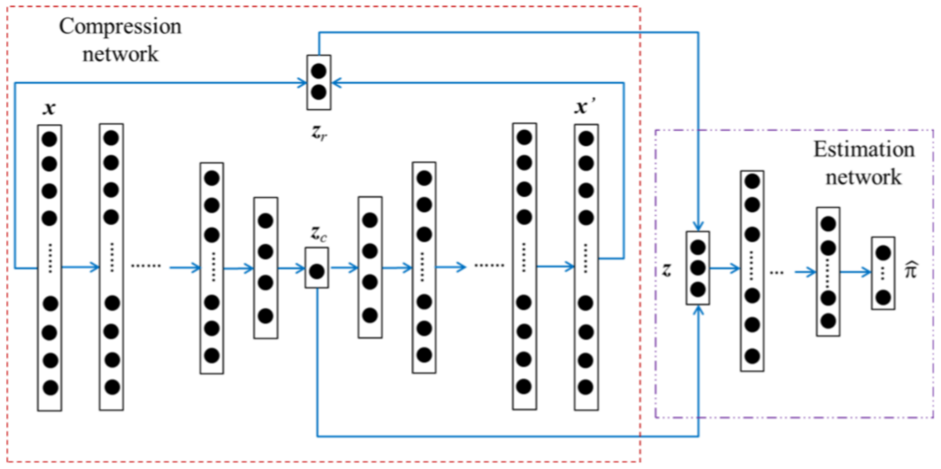

Figure 2: An overview on Deep Autoencoding Gaussian Mixture Model

Figure 4: Overview of the Deep Autoencoder and Gaussian Mixture Model.

References

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning.* MIT Press.

Hinton, G E, and R R Salakhutdinov. 2006. "Reducing the Dimensionality of Data with Neural Networks." *Science* 313 (5786): 504–7. https://doi.org/10.1126/science.1127647.

Kingma, Diederik P, and Max Welling. 2013. "Auto-Encoding Variational Bayes." http://arxiv.org/abs/1312.6114.

Ng, Andrew. 2011. "CS294A Lecture Notes, Sparse Autoencoders." Stanford.

Ranzato, Marc Aurelio, Christopher Poultney, Sumit Chopra, and Yann LeCun. 2007. "Efficient Learning of Sparse Representations with an Energy-Based Model." In *Advances in Neural Information Processing Systems 19 - Proceedings of the 2006 Conference*, 1137–44.

Rifai, Salah, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. 2011. "Contractive Auto-Encoders: Explicit Invariance During Feature Extraction." In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, 833–40. ICML'11. USA: Omnipress. http://dl.acm.org/citation.cfm?id=3104482.3104587.

Salakhutdinov, Ruslan, and Geoffrey Hinton. 2009. "Semantic Hashing." *Int. J. Approx. Reasoning* 50 (7): 969–78. https://doi.org/10.1016/j.ijar.2008.11.006.

Shafkat, Irhum. 2018. "Intuitively Understanding Variational Autoencoders." 2018. https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf.

Vincent, Pascal, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. "Extracting and Composing Robust Features with Denoising Autoencoders." In *Proceedings of the 25th International Conference on Machine Learning*, 1096–1103. ICML '08. New York, NY, USA: ACM. https://doi.org/10.1145/1390156.1390294.

Weiss, Yair, Antonio Torralba, and Rob Fergus. 2008. "Spectral Hashing." In *NIPS*.

Zhou, Chong, and Randy C. Paffenroth. 2017. "Anomaly Detection with Robust Deep Autoencoders." In *Proceedings of the 23rd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining*, 665–74. KDD '17. New York, NY, USA: ACM. https://doi.org/10.1145/3097983.3098052.

Zong, Bo, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. 2018. "Deep Autoencoding Gaussian Mixture Model for Unsupervised Anomaly Detection." In *International Conference on Learning Representations*. https://openreview.net/forum?id=BJJLHbb0-.