

An introduction to Autoencoders for Beginners

[ADVANCED](#)[AUTOENCODER](#)[DEEP LEARNING](#)[PROJECT](#)[PYTHON](#)

Introduction

Autoencoders in deep learning are unstructured learning models that utilize the power of neural networks to perform the task of representation learning. In the context of machine learning, representation learning means embedding the components and features of original data in some low-dimensional structure for better understanding, visualizing, and extraction of meaningful information autoencoder nlp. These low dimensional vectors can help us gain amazing information about our data such as how close two instances of the dataset are, finding structure and patterns in the dataset, etc.

This article was published as a part of the [Data Science Blogathon](#)

Table of contents

- [Current scenario of the industry](#)
- [Learning With Unlabeled Data](#)
- [Introduction to Autoencoders](#)
- [History of Autoencoders in papers](#)
- [Introduction to Variational Autoencoders](#)
 - [Regularization](#)
 - [Kullback-Leibler divergence](#)
- [VAE variants](#)
- [Applications of Autoencoders](#)
- [Frequently Asked Questions?](#)

Current scenario of the industry

In this [big-data](#) era, where petabytes of data are generated and processed by leading social networking sites and e-commerce giants, we are living in a world of data abundance. Our machine learning algorithms have only mainly exploited labeled datasets which are rare and costly. Most of the data generated are unstructured and unlabelled, so it is high time our machine learning community should focus on unsupervised learning algorithms and not just the supervised ones to unlock the true potential of AI and machine learning.

"If intelligence was a cake, unsupervised learning would be the cake, [supervised learning](#) would be the icing on the cake, and reinforcement learning would be the cherry on the cake" – Yann LeCunn

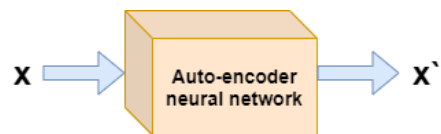
So why not eat the whole cake?

Learning With Unlabeled Data

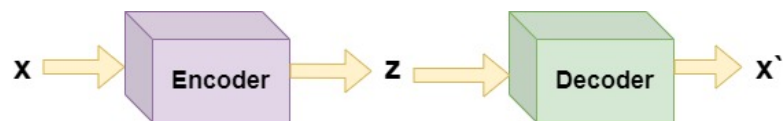
A representation of data is really a mapping. If we have a data point $x \in X$, and we have a function $f: X \rightarrow Z$ for some data space Z , then f is a representation. The new point $f(x) = z \in Z$ is sometimes called a representation of x . A good representation makes downstream tasks easier.

Introduction to Autoencoders

Autoencoders are also known as self-encoders are networks that are trained to reproduce their own inputs. They come under the category of unsupervised learning algorithms, in fact, some researchers suggest autoencoders as self-supervised algorithms as for a training example x , the label is x itself. But in a general sense, they are considered unsupervised as there are no classification or regression labels.

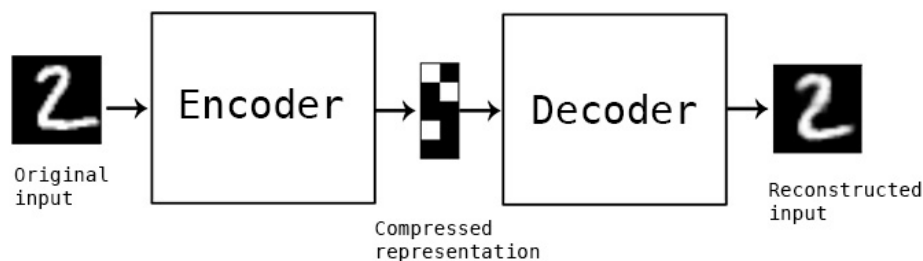


If an autoencoder does this perfectly, then the output vector x' is equal to the input vector x . The autoencoder is designed as a special two-part structure, the encoder, and the decoder.



$$AE = \text{Decoder}(\text{Encoder}(x))$$

The model train using the reconstruction loss which aims to minimize the difference between x and x' . We can define reconstruction loss as something like $MSE(x, x')$ if the inputs are a real value. The dimensionality of z is usually less than x , that's why autoencoders in deep learning are also called bottleneck neural networks. We are forcing a compressed knowledge representation of autoencoder nlp.



The encoder is used to map unseen data to a low dimensional Z and a good representation always focuses on the point that no important information was lost during compression. Autoencoders are just like Principle Component Analysis (PCA) which is itself a dimensionality reduction algorithm but the difference is PCA is linear in nature whereas autoencoders in deep learning are non-linear in nature due to neural net-based architecture. For a better understanding of the latent space, we can use an example where the observed variable x can be something like (number of people on beach, ice-cream sales, daily temperature) whereas the latent space z can be something like the tilt in Eath's axis (ie. season of the year) because

using the season information we can almost predict the number of visitors on the beach, ice cream sales, etc.

History of Autoencoders in papers

Below are research papers that are the first few to introduce AEs in the machine learning world :

- A learning algorithm for Boltzmann machines, DH Ackley, GE Hinton, TJ Sejnowski. Cognitive science, 1985. Describes a simple neural network trained by self-supervision.
- Learning representations by back-propagating errors, D. Rumelhart, Geoffrey E. Hinton, R. J. Williams. Nature, 1986. “We describe a new learning procedure, back-propagation, for networks of neuron-like units”.
- Connectionist learning procedures, GE Hinton. Machine learning, 1990. Describes the “self-supervised” bottleneck neural network.

Introduction to Variational Autoencoders

Variational autoencoder nlp are autoencoders in deep learning exploiting sampling technique and Kullback-Leiber Regularisation. The Variational autoencoders in deep learning aim to make the latent space smoother, i.e. a small change in x will lead to a small change in latent space z and a small change in z will lead to a small change in x . A latent space needs to be smooth with plausible points to be more effective and accurate and that is what VAE tries to achieve. In VAE, the encoder outputs not just z , but μ and σ . After which sampling operation chooses z from these parameters and as usual decoder takes z as before.

A good sampling technique provides a good reconstruction of data points, also a good reconstruction of points near to data points. The process ensures that every point that's close to the latent location where you encoded [the input x , ie z mean] can be decoded to something similar to $[x]$, thus forcing the latent space to be continuously meaningful. Any two close points in the latent space will decode highly similar images. Continuity, combined with the low dimensionality of the latent space, forces every direction in the latent space to encode a meaningful axis of variation of the data, making the latent space very structured and thus highly suitable for manipulation via concept vectors. The pseudo-code for sampling is detected below :

```
z_mean, z_log_variance = encoder(x)
z = z_mean + exp(z_log_variance) * epsilon
x_hat = decoder(z)
model = Model(x, x_hat)
```

In the VAE, we want the data to be distributed as a normal, in z -space. In particular, a standard multivariate normal, $N(0,1)$. When using the decoder, we can be confident that all such points correspond to typical x points. No “holes” because the encoder is working hard to compress the data, so it won't waste space.

Regularization

“The parameters of a VAE are trained via two loss functions: a reconstruction loss that forces the decoded samples to match the initial inputs, and a regularization loss that helps learn well-formed latent spaces and reduce overfitting to the training data.” – Chollet.

The regularisation loss asks the encoder to put the data into a normal distribution in the latent space.

Kullback-Leibler divergence

The Kullback-Leibler divergence $KL(p||q)$ is a statistical measure of dissimilarity between a pair of distributions, p , and q . So, it is a number that is large when p and q are dissimilar, and close to zero when they are similar.

The KL loss is a regularisation term in our VAE loss. As always, we can tune the regularisation by multiplying the KL by a scalar. If it is too strong our model will collapse and if too weak it will be equivalent to classic AE.

Remember: VAE = AE + Sampling + KL loss

- The sampling procedure samples from a multivariate Normal $N(\mu, \Sigma)$ to get each point z .
- The regularisation procedure adds a loss to push the latent distribution to be similar to a standard multivariate Normal $N(0,1)$.
- Usually, $\dim(z)$ is usually small compared to $\dim(x)$. How small? "As small as possible", without increasing reconstruction error too. As always, it depends on the downstream task. If $\dim(z) = 2$, we can easily visualize. But this is usually too extreme: can't reconstruct training data well much. If $\dim(z) > 2$, we might use t -SNE or UMAP for visualization.

VAE variants

- BetaVAE (stronger regularisation for disentangled representation).
- Contractive AE (aims for smoothness with a different regularisation).
- Conditional VAE (decoder maps $(z, c) \rightarrow x$, where c is chosen, e.g. c specifies which digit to generate and z specifies the style

Reference : [Keras Autoencoder](#)

Applications of Autoencoders

- Denoising images (Conv AE)
- Anomaly detection on time series (1D Conv AE)
- Network intrusion detection by anomaly detection (VAE Encoder only)
- Generation of video game levels and music (Conv VAE Decoder only).

Conclusion

At last, I feel autoencoder nlp and Variational autoencoders in deep learning are one the most powerful unsupervised learning technique that every data scientist should be aware of. Although these models have their own limitations like they require comparatively larger datasets for training etc.

I have attached the VAE code, there are two file [vae.py](#) which contains the encoder-decoder autoencoder model and [vae_visualize.py](#) that can be used to visualize the outputs.

Frequently Asked Questions?

Q1.What are autoencoders in unsupervised learning?

A. autoencoder nlp are competent learners in unsupervised learning who find important patterns in data without being told what to look for.

Q2.What is an autoencoder, and why is it called unsupervised learning?

A. An autoencoder is like a computer tool that learns independently without a teacher. It's called unsupervised learning because it learns directly from data without needing someone to say if it's right or wrong.

Q3.Are autoencoders trained without supervision?

A. Yes, autoencoder NLP learn by themselves without someone guiding them. They figure out the best way to represent data without needing labels or instructions.

The media shown in this article is not owned by Analytics Vidhya and are used at the Author's discretion.

Article Url - <https://www.analyticsvidhya.com/blog/2021/10/an-introduction-to-autoencoders-for-beginners/>



[Prakhar Gurawa](#)