Homework 1 : Unsupervised Deep Learning

João Ramos da Silva Tabuaço Freitas
Universitá Degli Studi di Padova
Student ID: 2009440

18 July 2022

Introduction

Unsupervised learning takes an agnostic approach in learning features in data. This distinguishes it from supervised learning, where models attempt to form a connection between an input and a ground-truth target. The performance of unsupervised learning models is normally assessed by how faithfully the data can be reproduced from the features learned.

Autoencoders and **generative-adversarial networks** (GANs) are two models used in unsupervised learning. In both cases, these models are composed by a perception module along with a reproduction module. These two work in tandem to achieve faithful reproduction of data with large number of features.

The goal of the autoencoder A is to learn to embed input data $\mathbf{x}$ in a latent space $\mathbb{R}^n$ of reduced dimensionality, and subsequently reproduce it. To achieve this, an encoder E transforms an input into a single vector in said latent space $z = E(x), z \in \mathbb{R}^n$. A decoder D then attempts to reproduce the original input $\mathbf{x}' = D(z)$ as faithfully as possible. This motivates the use of mean-squared error (MSE) to quantify the reconstruction error.

$$\{\hat{w}\} = \arg\min_{w} \frac{1}{n} \sum_{i=1}^{n} (\hat{\mathbf{x}}_i - \mathbf{x}_i)^2 \tag{1}$$

Of course, the autoencoder $A := D \circ E$.

Autoencoders with a latent space of lower dimension than the input space are said to be undercomplete, while those with a larger latent space are overcomplete. Evidently, the former performs dimensionality reduction, by learning only data's most prominent features. This proves useful in denoising input, for example. While the latter could have the capability of capturing larger detail from the inputs, its learning converges to the identity. In a denoising scenario, this means the output will be just as noisy as the input. This is simply a consequence of using reconstruction error as the learning criterion.

The performance of an undercomplete autoencoder can be compared with principal component analysis (PCA), or t-distributed stochastic neighbor embedding (t-SNE) on the input data. In fact, an autoencoder without non-linearities is expected to learn the structure of a PCA subspace. Naturally, non-linearity aids in capturing a more diverse set of features, which arguably endows the autoencoder with a larger descriptive power. To test this, its performance may be compared to t-SNE, which is well-established as a non-linear method of dimensionality reduction.

The structure of a GAN differs slightly from that of an autoencoder. It starts with a generator which produces samples $\mathbf{x} = g(\mathbf{z}, w_g)$ from a randomly generated latent vector $\mathbf{z}$, where w_g are the generator parameters. These samples are then fed into the discriminator, whose goal is to observe an input, and reach a conclusion $y = d(\mathbf{x}, w_d) = d(g(\mathbf{z}, w_g), w_d)$ about said input (with w_d being the discriminator parameters). The learning task can be formulated as a zero-sum game solved by a minimax decision.[1] The discriminator observes a payoff $v(g, d)$, while the generator receives $-v(g, d)$. Thus, the goal is to find g^* such that:

$$g^* = \arg\min_g \max_d v(g, d) \tag{2}$$

Normally, the choice of $v(g, d)$ is:

$$v_{\text{mm}}(g, d) = -\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log(d(\mathbf{x}))] - \mathbb{E}_{\mathbf{x}\sim p_g}[\log(1 - d(\mathbf{x}))] \tag{3}$$

In other words, the aim of the minimax game is for the discriminator to maximize its payoff by classifying fake/real inputs correctly, while the generator attempts to minimize the discriminator's payoff by tricking it into believing that the inputs are real.

One issue with the minimax game formulation is that the gradient quickly diverges away from $d(\mathbf{x}) = 1/2$. This can be addressed by modifying the generator loss, as is done in non-saturating GAN (NSGAN) formulation.[2] In this scenario, the discriminator loss is v_{mm}, while the generator loss is:

$$v_{\text{NS}}(g, d) = -\mathbb{E}_{\mathbf{x}\sim p_g}[\log(d(\mathbf{x}))] \tag{4}$$

Both the minimax and the non-saturating formulations will be explored.

Methods

The FashionMNIST dataset is used to explore both the autoencoder and the GAN, with each image having shape $(1, 28, 28)$. The set used for training consisted of 60000 equally

[1] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014c). Generative adversarial networks. In NIPS'2014.

[2] Lucic, M., Kurach, K., Michalski, M., Bousquet, O., Gelly, S., (2018). Are GANs Created Equal? A Large-Scale Study. In NIPS'2018.

distributed samples, which was sectioned to serve the purpose of validation as well. The test set consists of 10000 samples with the same properties.

The autoencoder architecture consists of three convolutional layers, followed by two fully connected layers. Regularization is implemented through batch normalization on the convolutional layers, and L2 penalty on the loss. The number of convolutional filters, number of fully-connected neurons, optimizer algorithm, and learning rate are tuned using Optuna, which attempted 20 sets of parameters, training each model over five epochs. The latent space dimension is also tuned, but kept below 15 to maintain the model undercomplete. However, dimensionality reduction is explored using latent dimension $\dim(z) = 2$ for ease of visualization. The convolutional kernel sizes, strides and padding are kept constant to preserve the same output shapes in all trials. While different values for these could be attempted, preliminary attempts showed no discernible difference in training convergence or model performance.

Upon tuning the model architecture, the model was validated using a K-Fold cross-validation strategy, splitting the dataset in six sequential folds and performing training over 20 epochs. The performance was assessed based on the reconstruction error on the test set.

To test the expressive power of the model, a convolutional neural network was trained to classify the set of latent vectors corresponding to the input data. The network's learning curve was compared to a similar network trained on the unaltered dataset, over 50 epochs.

Finally, a new encoder's dimensionality reduction was visually compared to PCA and t-SNE. A perfect encoder would be able to spatially separate the data points in the latent space. Of course, some confusion is expected between some classes of similar-looking items, both for the encoder and the other methods.

The creation and training of the GAN posed several challenges. The level of complexity of the model discouraged hyperparameter tuning. This was because the Optuna framework would have to explore the hyperparameter space based on two different metrics (generator and discriminator loss). The set of parameters which would improve the generator performance could hinder the discriminator's, and vice-versa. Because of this, automated tuning was not performed. Furthermore, the training process revealed itself demanding from a computational standpoint. As a result, only one model was trained without cross-validation. Given the well-known regularity of the FashionMNIST dataset, this is acceptable, as there are no large disparities in different sections of the dataset.

Both components of the GAN were endowed with four convolutional layers, and no fully-connected section. The number of filters, kernel sizes, strides, and padding were chosen in order to have the generator outputting an image with the same dimensions as the FashionMNIST inputs, and the discriminator outputting a single scalar value.

Results

The best architecture found for the autoencoder is described in Table 1. The K-fold results (Figure 1) show little variability in the learning curves, which is expected by the fairly regular nature and even distribution of class labels in the dataset.

While the training loss experiences larger variance, the trend is similar for training and validation across all folds. The convergence of training and validation loss close to 20 training epochs indicates the model is well-regularized and not overfitting.

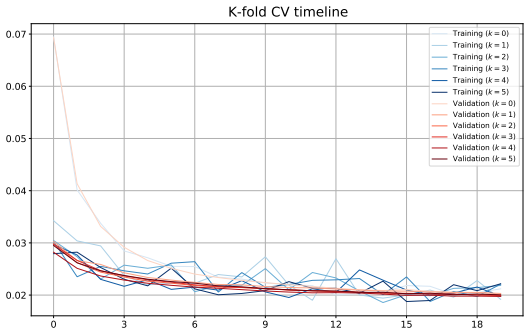

Figure 1: K-fold learning curves.

The denoising capabilities of the autoencoder were tested by inputting an image with different levels of noise. One sample image is reconstructed (Figure 2), and the loss was shown to improve in all extents of noise tested. However, this is not always the case, and generally depends on the sample used. Samples with a larger number of blackened pixels on their own are less likely to see an improvement. This is because the choice of blackened pixels is uniformly random, so the noisy sample might have stayed very similar to the original. The model struggled more to reconstruct samples when Gaussian noise was added. This could be attributed to the lack of noisy samples in the training process, and was left out of the analysis.

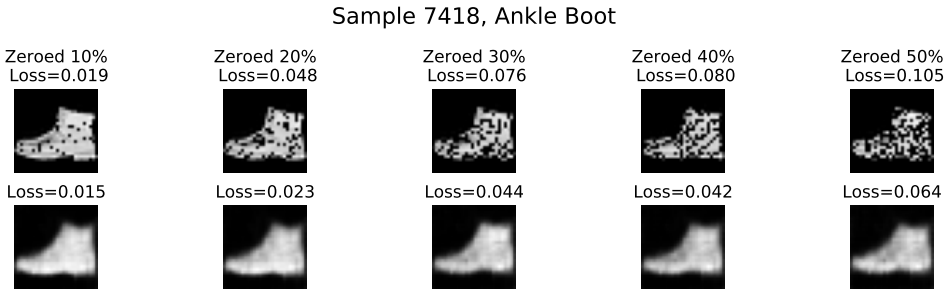

Figure 2: Denoising samples

Layer	No. channels Neurons	(Kernel, Strides, Padding)	Activation	Batch Normalization
conv1	7	(3, 2, 1)	ReLU	Yes
conv2	10	(3, 2, 1)	ReLU	Yes
conv3	16	(3, 2, 0)	ReLU	Yes
Lin1	32	N/A	ReLU	N/A
Latent	3	N/A	N/A	N/A
Lin1	32	N/A	ReLU	N/A
deconv1	16	(3, 2, 0)	ReLU	Yes
deconv2	10	(3, 2, 1)	ReLU	Yes
deconv3	7	(3, 2, 1)	ReLU	Yes

Table 1: Autoencoder architecture. The no. of channels in conv[i] corresponds to output channels, while it corresponds to input channels in Deconv[i].