Investigation of latent space structure

Jacopo Lancione
Università di Torino, jacopo.lancione@edu.unito.it

The data is often supposed to have a structure, namely the manifold hypothesis. This project tries to explore the possibility of part of this structure being encoded in the latent representation developed by different kinds of autoencoders.

1 Question and insights

This work revolves around latent representations of data and how far the insight of the manifold hypothesis can be pushed. In other words, is there really a connection between the latent representation built by autoencoders and the supposed data manifold?

The intuition suggests that within the space of the features, the actual meaningful data lies on an embedded manifold. So the data is expected to have some kind of structure. To inspect this supposed structure, it is meaningful to use data that already suggests part of it. Then verify if this is reflected in the representation developed by the autoencoder. For the sake of concreteness, let us develop straight away the example at the basis of this work. The dataset used is MNIST, the feature space is the 28×28-dimensional space of all possible greyscale images. The data are handwritten digits and some structure of the data is suggested by the labels, i.e. the digits. Now the key argument: by moving on the supposed data manifold, one would expect to reach all the instances of the same digit without crossing to another digit. Therefore, different digits should be clustered on the learnt representation of such manifold, since starting from a data point labelled with digit 6, it would be awkward to be forced to go through a region of 2s to reach another instance of digit 6. This is a structure one would expect to be preserved in a somewhat faithful representation of the data manifold.

This leads to a precise task for the analysis: inspect clusters of digits in the latent representations learned by the models. Better representations will have better separated clusters. These spaces will then be the best candidates for the approximation of the data manifold.

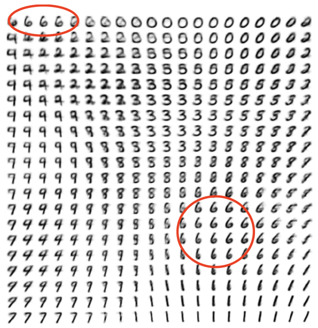


Fig. 1: *2-dimensional latent representation developed with a variational autoencoder [1]. The digit 6 appears at least in two different disconnected regions.*

The proposed argument infers that representations such as fig. 1 have little to do with the supposed data manifold, since in the encoding process they lose part of the structure of the data, which could result in an inefficient latent code.

2 Methods

To have a better understanding of the analysis, I decided to fix the latent dimension of all the models to 2. There are two main reasons for this: first, I wanted to keep a solid understanding of the latent space by being able to visualize it without any sort of projection; second, because the dimension of the data manifold is unknown, but at least it has to take into account rotations and stretches of the digits' images. These are supposed to be the main transformations that map data points into data points of the same class. Thus, as a rough approximation, 2 is supposed to be a sufficient dimension to preserve at least the major structures in the data

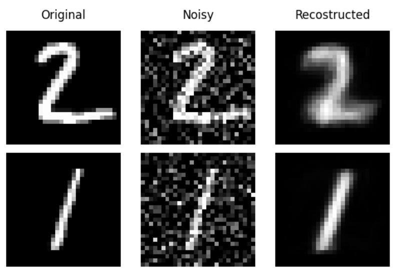


Fig. 2: *Denoising autoencoder recostructed images*

Fully connected layers	
Encoder	$28 \times 28 \mapsto 128 \mapsto 32 \mapsto 2$
Decoder	$2 \mapsto 32 \mapsto 128 \mapsto 28 \times 28$

Tab. 1: *Autoencoder architecture summary (the denoising case uses the same). Encoder and decoder weights are not tied. Training specifics: Adam as optimizer with learning rate $lr = 10^{-2}$, weight decay $= 10^{-6}$. For the denoising version, the gaussian noise factor is 0.3*

of the single class. But, to tackle the question posed initially, at least 2 digits are needed to study how well the latent representations learnt keep them separated.

To sum up, I considered 2 labels at a time from dataset MNIST; then trained a model able to map the data to a 2-dimensional space, and at last studied the distribution of new data in the latent representation. The analysis was carried out for three different models: a PCA algorithm, an autoencoder, and a denoising autoencoder (fig. 2). To compare the latent representations, the metric chosen is the silhouette s score [3].

2.1 Autoencoder architecture

The two kinds of autoencoders considered, normal and denoising, share both the architecture and the training procedure[1]. Since the network is forced to compress the input to a very short code, there is no need to ask for sparsity on the latent neurones. Trying to impose constraints on only these two neurones worsens the performance, so the loss function is the MSE

$$J(x, x_{\text{rec}}) = \frac{1}{2m} \|x - x_{\text{rec}}\|_F^2,$$

with the Frobenius norm since the images are matrices. This choice to avoid regularization goes on to the encoder and decoder, summarized in table 1, whose weights are not tied. Again, the latent space dimensionality is already a strong constraint imposed on the net that is asked to learn an efficient representation. Moreover, I observed that adding other hidden layers for a more gradual compression does not perform as well as the architecture proposed in table 1.

[1] The code to reproduce all the results is hosted in the repository `https://github.com/jlancione/DeepLearningProject`

Model evaluation was carried out using the validation loss as metric. All the preceding reasoning is based on this choice.

2.2 Activation function

Between the fully connected layers, the preferred activation function is tanh. With the same architecture, it performs slightly better than ReLU, referring to the validation loss, but it is able to provide higher quality representations with a better separation of the labelled data. The reason lies in the hard cutoff of the ReLU that causes most of the data to gather around the origin, making the clustering task harder. This proved to be especially true in the case of hard to distinguish couples of digits, for example 1 and 7, where the network with ReLU reconstructs just a cloudy average digit, while the one with tanh manages not only to reconstruct them but also to cluster them.

2.3 PCA

The algorithm is implemented as a linear autoencoder, since it is known that PCA is the best linear autoencoder, provided to impose tied weights. Therefore, the code is the same as the autoencoder implementation, removing activation functions and hidden layers while keeping the 2-dimensional latent space. This leads to the same latent space representation as the classical implementation of PCA, but written in a different vector basis, not the standard principal component basis found diagonalizing the covariance matrix. The principal components can still be worked out [2] but it is not the focus here; what is relevant is the latent space, which is the same.

So even if it is an algorithm, the weights were discovered through the same training procedure. However, the loss saturates after a few epochs, always to the same value fixed by the couple of digits chosen, showing that there is an optimal solution to the problem, even if there are different sets of weights corresponding to different bases of the same space. The reconstruction of the images has lower quality than the autoencoder. They are more blurred and have higher inaccuracy even if the validation loss is comparable.

2.4 Silhouette score

The silhouette s score [3] provides a way to quantitatively argue about the goodness of the clusterization of some data. For each data point i compute the mean distance $a(i)$ to the other points in the same cluster and the mean distance b to points in the neighbor cluster, then the score is defined as

$$s_i = \begin{cases} 1 - \frac{a_i}{b_i} \text{ if } a_i < b_i \\ \frac{b_i}{a_i} - 1 \text{ if } a_i \geq b_i \end{cases} .$$

Thus $s \in [-1, 1]$, with $s \approx 1$ if the point can be classified without doubt and $s = 0$ if it is on a boundary between clusters. To compare clustering procedures on the same dataset, one relies on the average s_{avg} or graphically, as in fig. 3, as will be discussed in the next section.

Here it will be used in a somewhat different way, since the number of clusters and the partitioning of the data is known. The issue lies in the way it is represented. Therefore, this score will be a measure of the quality of the transformation carried out by the encoder.

Unfortunately, this score must not be interpreted as some kind of invariant across the optimal encoding, which would identify the data manifold. The average silhouette score on the original 28×28 data is $s_{\text{orig}} = 0.11$ which stands for a very weak clustering, while we expect the points to be well organized on the data manifold. This is readily interpreted through the lens of the curse of dimensionality, which suggests that each data point is isolated from all the others. Therefore, one should not expect to recognise a clustering structure in the original representation. In the language of the s score this means $s \approx 0$, i.e. all the data is as far from its own cluster as it is from the neighboring one. In other words, in the original representation every data point is located near the boundary between clusters.

3 Results

The silhouette plots in fig. 3 are horizontal bar diagrams with the s score for every data point ordered vertically from the highest. They are a graphical way to have an intuition of the cluster. If the silhouette, meaning the shaded region, is wide $s \approx 1$ and those points are in the bulk of the cluster, while if it gets narrow $s \approx 0$ and they are near the boundary. When it flips to the left it means that they are more easily identified as points belonging to the neighbor cluster.

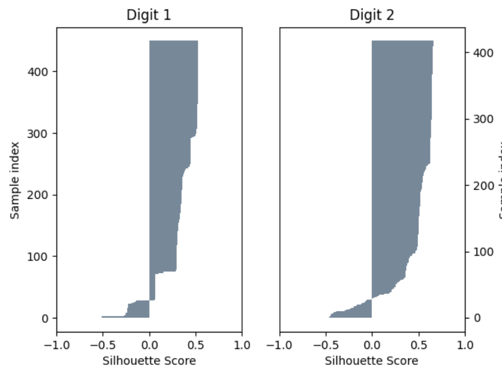
The plots of the form in fig. 4 are used to visualise the latent space and how the data mapped onto it by the model. By confronting these two different kinds of plots, one gets a sense of what it means for a silhouette to be narrow. In fig. 4 digit 1 has a narrower silhouette in training A, seen from the latent space point of view, this reflects to harder to distinguish points in the representation in fig. 4a, especially in the lower left quarter.

Plots of these kinds were produced and studied side by side, repeating for the three different models considered. I also tried with different couples of digits. During this analysis, I used only the validation data in order to avoid the pitfall of making assertions on the model using the same data it was trained with. This is what I have found:

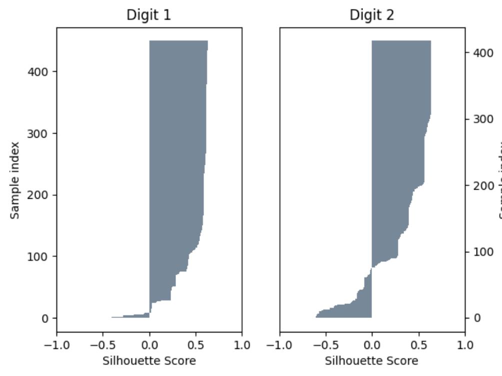
- The organization of the latent space is extremely sensitive to initial conditions of the training, producing significantly different representations from one training to another, with all the hyperparameters kept fixed. This is particularly evident by comparing figs. 4a and 4b, where points are positioned in widely different ways.

- The corresponding silhouettes in figs. 3a and 3b do not seem as different as the corresponding representations. This qualitative statement relies on the score s_{avg}. This situation is not an isolated exception, as one might expect. The variability on the silhouettes seems to be less pronounced with respect to the corresponding latent space plots. Nonetheless, there were cases with higher scores, up to $s_{\text{avg}} = 0.60$ meaning quite a strong clustering in the latent space.

- For different couples of digits the results are analogous, apart from those hard to distiguish such as 4 and 9 or 1 and 7, where the



(a) *Training A:* $s_{\text{avg}} = 0.41$ $\text{Loss}_{\text{val}} = 0.034$



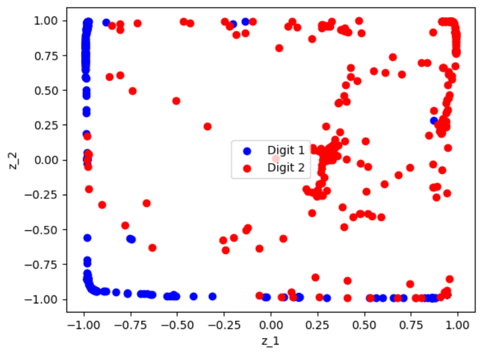
(b) *Training B:* $s_{\text{avg}} = 0.44$ $\text{Loss}_{\text{val}} = 0.034$

Fig. 3: *Silhouette plots for two different trainings of the denoising autoencoder, with all the hyperparameters kept fixed.*

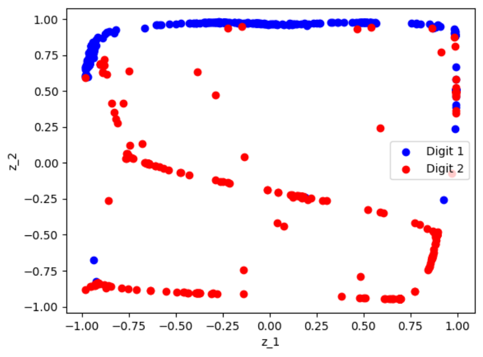
denoising autoencoder fails to reconstruct the digits. In the cases where both the normal and denoising autoencoders manage to reconstruct the image, there is no difference in the clustering that manifests over the variability to initial conditions.

- There are cases where the variability with respect to initial conditions manifests in a peculiar way. They show a clear-cut imbalance between the width of the two silhouettes, but it happens both ways as if the model could not find a trade-off and resorts to clustering sharply only one digit at random.

- PCA produces a latent space where the clustering scores $s_{\text{avg}} = 0.47$, a value within the variability observed in the autoencoders. This number refers to the couple of digits 1 and 2, and it only changes a little when considering other couples of digits.



(a) *Training A:* $s_{\text{avg}} = 0.41$ $\text{Loss}_{\text{val}} = 0.034$



(b) *Training B:* $s_{\text{avg}} = 0.44$ $\text{Loss}_{\text{val}} = 0.034$

Fig. 4: *Validation data distribution in the latent space with coordinates* (z_1, z_2) *for two different trainings of the denoising autoencoder with all the hyperparameters kept fixed. These trainings correspond to the ones in fig. 3.*

4 Conclusions

The autoencoders considered can endow the latent space with some structure at least in the form of clusters. This points in the direction of the data manifold but the variability observed with respect to initial conditions suggests that it is merely the autoencoder trying to find an efficient representation. There is no evidence of any sort of convergence to some preferred representation, even on the same model trained with the same hyperparameters. Therefore, the insight is that this kind of encoding procedure tends to provide the latent space with a structure that has little to do with the expected data manifold. On the flip side, it is evident that there are many different representations, and there is no hint of them being equivalent.

As far as this analysis goes, there is no manifest advantage in choosing a normal autoencoder or the denoising version. More surprisingly, their performance in the explored task shows no significant difference from the linear encoding given by PCA. Recall that the task at hand is providing the latent space of some cluster structure, not the ability to reconstruct the images, where PCA has an inferior performance compared to the non-linear models.

The silhouette score s_{avg} gives a practical graphical representation for studying clustering problems. Moreover, in the example studied, it may show a smaller variability than the underlying latent representation, since it doesn't seem to depend as strongly on the initial conditions of the training. This comment is merely qualitative, giving an account of what the repeated experiments seemed to suggest.

It was found that hard cut-offs such as the one in ReLU can hinder the ability of the network to provide the latent space of the cluster structure considered, preventing the model building an efficient representation of the data.

5 Future developments

The question posed cannot possibly be reduced to the study conducted here, which is inherently flawed in many aspects. One such limitation regards the metrics used to choose the model for the analysis. The loss on the validation set does not take into account the cases where the reconstructed image is more similar to the other class than the original label. This criterion is arguably more relevant than the validation loss itself, since the whole analysis revolves around the latent space organization of the data and a wrong *recostruction of the label* (this is the metric proposed) is expected to imply a mispositioning in the latent representation.

Another critical aspect of the problem is the dimensionality of the latent space. The choice of taking it 2-dimensional is due to the argument that each digit has to live at least in a 2-dimensional space, since it has to take into account rotations and stretches of the image. This is of course an approximation, since for a single label there is a higher variability that cannot be reduced to the aforementioned transformations. Therefore, a proper study across different latent space dimensions is certainly encouraged since one might expect the data manifold to be approximated better in a higher-dimensional latent space.

One further interesting discussion would relate the machine learning methods explored here with more controlled algorithms of manifold identification. There exist many different non-linear methods of manifold approximation that rely on geometrical arguments. An interesting one is named *Manifold Sculpting* [4], whose objective is to transform the data while trying to preserve local relations, namely distances and angles between neighbor points. This has been part of the idea at the outset of the original explorative analysis of different representations of the latent space. In the repository `https://github.com/jlancione/DeepLearningProject` there is a single-core implementation[2] in PyTorch of such algorithm, but its computational complexity made it unfeasible to conduct a proper analysis with the available resources.

References

[1] Kingma, D. P. & Welling, M. Auto-Encoding Variational Bayes. Preprint at https://doi.org/10.48550/arXiv.1312.6114 (2022).

[2] Plaut, E. From Principal Subspaces to Principal Components with Linear Autoencoders. Preprint at https://doi.org/10.48550/arXiv.1804.10253 (2018).

[3] Rousseeuw, P. J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics 20, 53–65 (1987).

[4] Gashler, M., Ventura, D. & Martinez, T. Iterative Non-linear Dimensionality Reduction with Manifold Sculpting. in Advances in Neural Information Processing Systems vol. 20 (Curran Associates, Inc., 2007).

[2] Pseudocode in the paper [4] and reference implementation in the repository `https://github.com/Gabriele-Codega/Manifold_Sculpting`.