ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2. Available from http://www.i6doc.com/en/.

Visualization of the Feature Space of Neural Networks

Carlos M. Alaíz, Ángela Fernández and José R. Dorronsoro *

Dpto. de Ingeniería Informática & Instituto de Ingeniería del Conocimiento
Universidad Autónoma de Madrid, 28049 Madrid - Spain

Abstract. Visualization of a learning machine can be crucial to understand its behaviour, specially in the case of (deep) neural networks, since they are quite difficult to interpret. An approach for visualizing the feature space of a neural network is presented, trying to answer to the question "what representation of the data is the network using to make its decision?" The proposed method gives a representation of the space where the network is tackling the problem, reducing it while respecting the linearity of the model. As shown experimentally, this technique allows to study the evolution of the model with respect to the training epochs, to have a representation of the data similar to the one used by the neural network, and even to detect groups of patterns that behave differently.

1 Introduction

Given the current interest in the interpretability of Machine Learning models, visualization has become an even more crucial issue. In particular, there is an interest on visualizing machine learning models, either for interpreting them or just for model inspection. In that case the visualization has a supervised component, since the trained model has information about the target.

This work proposes an approach to understand how a Neural Network (NN) is transforming the data, in order to analyse its behaviour and its structure, and hopefully to use this knowledge to improve it. This method covers any network, independently of the complexity of their architectures. The main idea is to visualize the feature transformation that the model applies over the data, representing the points in the embedded space where the decision is taken. This can be useful for understanding the structure of the final model, but also to see its evolution along the different training epochs.

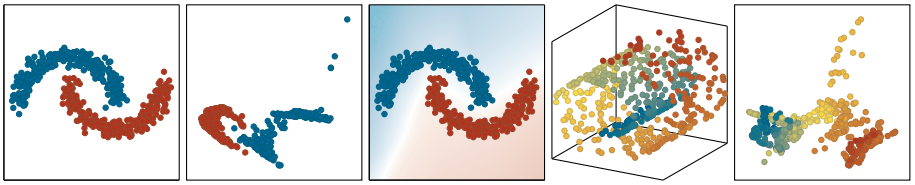
The paper is organized as follows: Sec. 2 briefly reviews the state of the art in NN visualization, whereas the proposed method is explained in Sec. 3. Section 4 includes numerical experiments, and some conclusions are given in Sec. 5.

2 Neural Networks and Visualization

Artificial NNs are classical non-linear learning machines that have become state-of-the-art techniques for solving many real problems. Their basic processing

*With partial support from the European Regional Development Fund and from the Spanish Ministry of Economy, Industry, and Competitiveness, project TIN2016-76406-P (AEI/FEDER, UE). Work supported also by UAM–ADIC Chair for Data Science and Machine Learning. We also acknowledge the use of the facilities of Centro de Computación Científica (CCC) at UAM.

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2. Available from http://www.i6doc.com/en/.



(a) `moons`: Orig. (b) `moons`: PCA. (c) `moons`: Pred. (d) `roll`: Orig. (e) `roll`: PCA.

Fig. 1: Original `moons` and `roll` datasets, PCA projections of the transformed feature spaces and predicted output over the original space (only for `moons`).

units are the neurons, that compute their outputs as weighted linear combinations of the inputs plus an activation function. In Feed-Forward Neural Networks (FFNNs), the neurons are organised in layers, where the output of each layer is used as input of the next one. Any NN in which the last layer is composed by a single logistic (for classification) or linear (for regression) unit can be split into a transformation of the feature space $\varphi : \mathbb{R}^d \to \mathbb{R}^D$, that take places in the hidden layers, and a linear classification or regression model, $\{\mathbf{w}, b\} \in \{\mathbb{R}^D, \mathbb{R}\}$, performed by the output unit.

Regarding visualization methods that represent how a NN sees the data, the most interesting works are naturally applied to images, where there are mainly two approaches. The first one is to find an input image that gives more evidence for/against one particular class, e.g. in [1] two tools for visualizing convolutional NNs are applied to images and videos, based on making a regularized optimization of the activation patterns in the input space; in [2] the idea of plotting the activations to check the structures that excite the map also appears. The second one is to visualize how the network answers when it receives an image, exploring a particular classification made by the network. In this case it is particularly relevant the work in [3], where difference analysis is used to highlight the parts of the image that have more evidence for or against one class. Finally, a recent approach [4] shares the goal of this work, looking for a visualization of the data space that provides insights into the NN, although using a non-linear embedding.

It should be noted that a trivial representation of any machine learning model can be done when the initial dimension is small enough ($d < 3$). In that case the data can be directly plotted in the original space, and the prediction can be computed over a grid, as shown in Fig. 1, where the original two-dimensional dataset of Fig. 1a is depicted jointly with the surface of the prediction of the model in Fig. 1c. The main drawback of this approach is that it can only be applied to problems with one or two features.

3 Visualization of the Feature Space

Unlike the approaches described above, the method proposed next provides a visualization of the data after the transformation that takes place inside the

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2. Available from http://www.i6doc.com/en/.

NN, but preserving this implicit embedding. Let $\mathbf{x}_i \in \mathbb{R}^d$, $i = 1, \ldots, N$, be the input patterns and $\mathbf{X} \in \mathbb{R}^{N \times d}$ the data matrix. Considering the usual NNs composed by a feature transformation plus a linear model, once the initial data matrix $\mathbf{X} \in \mathbb{R}^{N \times d}$ has been transformed into $\mathbf{\Phi} \in \mathbb{R}^{N \times D}$ (where the i-th row of $\mathbf{\Phi}$ corresponds to the i-th transformed pattern, $\boldsymbol{\phi}_i = \varphi(\mathbf{x}_i)$), the visualization reduces to represent a linear model $\{\mathbf{w}, b\}$ with the corresponding D-dimensional data $\mathbf{\Phi}$. Although this visualization is trivial if $D \leq 3$, in the general case some dimensionality reduction should be applied, ideally maintaining the original structure to keep the linear relationship between the input and the prediction.

Therefore, the objective is to reduce the extended feature space from D dimensions to only two/three components that can be accurately depicted. Since the visualization method should be somehow compatible with the NN, and the prediction of the NN is linear in $\mathbb{R}^D$, the transformation proposed here will also be linear. Hence, the dimensionality reduction will be defined by a projection matrix $\mathbf{P} \in \mathbb{R}^{d' \times D}$ (typically $d' = 2$ or 3). Moreover, and in order not to maintain redundant information, the rows of $\mathbf{P}$ will be assumed to be orthogonal, and with norm equal to one to preserve the original metric as much as possible.

The standard approaches in such a context do not consider in any way the linear model $\{\mathbf{w}, b\}$ on top of the NN. For example, although Principal Component Analysis (PCA; [5]) is designed to preserve as much variance of the data as possible, the prediction of the NN will not be linear in the PCA-reduced space. Hence, it will not be possible to depict the prediction in the reduced space using a grid since each point of the grid corresponds to a subspace of points of the original space, and each point of that subspace can have a different prediction of the model. This effect is illustrated in Figs. 1b and 1e, where the feature space is reduced with PCA for the classification and regression datasets of Figs. 1a and 1d (described in detail in Sec. 4). The target, represented by the colours of the points, is not linear despite the high precision of the NNs; this means that, although both NNs have successfully projected the data so that the problem is (almost) linear, the visualization using PCA does not allow to appreciate it.

A dimensionality reduction technique that respects the linearity of the prediction is thus needed. Such a technique is defined thanks to the next proposition.

Proposition 1. *Let $\mathbf{P} \in \mathbb{R}^{d' \times D}$ be a projection matrix from $\mathbb{R}^D$ to $\mathbb{R}^{d'}$ (typically $d' \ll D$) and let $\{\mathbf{w}, b\} \in \left\{\mathbb{R}^D, \mathbb{R}\right\}$ be a linear model on $\mathbb{R}^D$. The projection $\mathbf{P}$ preserves the linearity of the linear model if and only if $\mathbf{w} \in \operatorname{span}(\mathbf{p}_1, \ldots, \mathbf{p}_{d'})$, with $\mathbf{p}_i \in \mathbb{R}^D$ the i-th row of $\mathbf{P}$.*

Proof. The projection $\mathbf{P}$ preserves the linearity of a linear model if there exists an equivalent linear model in the reduced space such that the prediction of the original model for any $\boldsymbol{\phi} \in \mathbb{R}^D$ is equal to the prediction of the equivalent model for $\mathbf{P}\boldsymbol{\phi} \in \mathbb{R}^{d'}$, i.e., if there exists a hyper-plane $\mathbf{w}' \in \mathbb{R}^{d'}$ such that $\forall \boldsymbol{\phi} \in \mathbb{R}^D$, $\boldsymbol{\phi}^\intercal \mathbf{w} = (\mathbf{P}\boldsymbol{\phi})^\intercal \mathbf{w}' = \boldsymbol{\phi}^\intercal \mathbf{P}^\intercal \mathbf{w}'$. Since this has to be satisfied for any vector $\boldsymbol{\phi} \in \mathbb{R}^D$, in particular it has to be satisfied for the D vectors of the canonical basis of $\mathbb{R}^D$, $\mathbf{e}_i^\intercal \mathbf{w} = \mathbf{e}_i^\intercal \mathbf{P}^\intercal \mathbf{w}'$. Stacking all these equalities together leads to the matrix equality $\mathbf{I}\mathbf{w} = \mathbf{I}\mathbf{P}^\intercal \mathbf{w}'$, with $\mathbf{I} \in \mathbb{R}^{D \times D}$ the identity matrix. Therefore, $\mathbf{w} = \mathbf{P}^\intercal \mathbf{w}' = \sum w'_i \mathbf{p}_i$, and hence $\mathbf{w} \in \operatorname{span}(\mathbf{p}_1, \ldots, \mathbf{p}_{d'})$. □

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2.
Available from http://www.i6doc.com/en/.

Algorithm 1 Visualization of the Feature Space of a Neural Network.

Require: Feature Mapping $\varphi : \mathbb{R}^d \to \mathbb{R}^D$, Output Weights $\{\mathbf{w}, b\} \in \left\{\mathbb{R}^D, \mathbb{R}\right\}$, Original Data $\mathbf{X} \in \mathbb{R}^{N \times d}$
Ensure: Embedded Data $\mathbf{\Psi} \in \mathbb{R}^{N \times d'}$
$\mathbf{\Phi} \leftarrow \varphi(\mathbf{X})$ ▷ Transformed data.
$\mathbf{p}_1 \leftarrow \mathbf{w}/\|\mathbf{w}\|$ ▷ First direction of projection.
$\boldsymbol{\psi}_1 \leftarrow \mathbf{\Phi}\mathbf{p}_1$ ▷ First component of the projected data.
$\mathbf{\Phi}_c \leftarrow \mathbf{\Phi} - \boldsymbol{\psi}_1 \mathbf{p}_1^\intercal$ ▷ Removal of the projected information.
for $i = 1, \ldots, d' - 1$ **do** ▷ PCA over $\mathbf{\Phi}_c$ to obtain $\mathbf{\Psi}_c$.
  $\mathbf{u}_{c,i} \leftarrow \textsc{EigVec}(\mathbf{\Phi}_c^\intercal \mathbf{\Phi}_c, i)$ ▷ Eigenvector with i-th largest eigenvalue.
  $\boldsymbol{\psi}_{i+1} \leftarrow \mathbf{\Phi}\mathbf{u}_{c,i}$ ▷ Component $i+1$ of the projected data.
end for
return $\mathbf{\Psi}$

Given Prop. 1 and up to rotations, the first component on the reduced space should be the projection over the direction of $\mathbf{w}$. Hence, the first row of $\mathbf{P}$ is defined as $\mathbf{p}_1 = \frac{\mathbf{w}}{\|\mathbf{w}\|}$, and the first dimension of the reduced space is $\boldsymbol{\psi}_1 = \mathbf{\Phi}\mathbf{p}_1$. Since the linearity of the prediction is already guaranteed thanks to $\boldsymbol{\psi}_1$, the other components $\boldsymbol{\psi}_2, \ldots, \boldsymbol{\psi}_{d'}$ can be designed to maximize the variance retained. Hence, they have to be orthogonal to $\mathbf{p}_1$ and they have to maximize the variance, and this is precisely what PCA does when applied to $\mathbf{\Phi}_c = \mathbf{\Phi} - \boldsymbol{\psi}_1 \mathbf{p}_1^\intercal$ (the data once $\mathbf{p}_1$ has been projected out), as stated in the next proposition.

Proposition 2. *Let $\mathbf{\Phi} \in \mathbb{R}^{N \times D}$ be a data matrix, and let $\mathbf{p}_1 \in \mathbb{R}^D$ be a projection direction with $\|\mathbf{p}_1\| = 1$. Then the projection $\mathbf{P}_c$ corresponding to PCA applied over $\mathbf{\Phi}_c = \mathbf{\Phi} - \boldsymbol{\psi}_1 \mathbf{p}_1^\intercal$, with $\boldsymbol{\psi}_1 = \mathbf{\Phi}\mathbf{p}_1$, is a linear projection orthogonal to $\mathbf{p}_1$ that maximizes the variance retained when applied to $\mathbf{\Phi}$.*

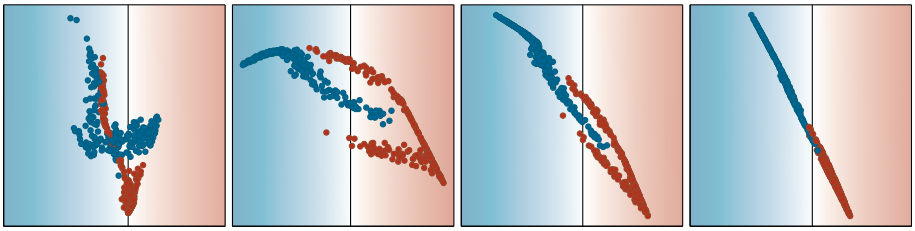
Proof. Since $\mathbf{\Phi}_c \mathbf{p}_1 = \mathbf{\Phi}\mathbf{p}_1 - \boldsymbol{\psi}_1 \mathbf{p}_1^\intercal \mathbf{p}_1 = \boldsymbol{\psi}_1 - \boldsymbol{\psi}_1 \|\mathbf{p}_1\|^2 = \mathbf{0}$, then $\mathbf{p}_1$ is an eigenvector with zero eigenvalue of the covariance matrix $\mathbf{\Phi}_c^\intercal \mathbf{\Phi}_c$, and hence this component will not be selected by PCA, and $\mathbf{P}_c$ is orthogonal to $\mathbf{p}_1$. In order to see that it maximizes the variance, let $\mathbf{P}'_c$ be a different projection orthogonal to $\mathbf{p}_1$, i.e., $\mathbf{P}'_c \mathbf{p}_1 = \mathbf{0}$. Since $\mathbf{\Phi}$ and $\mathbf{\Phi}_c$ only differ in a component on the direction of $\mathbf{p}_1$, $\mathbf{\Phi}(\mathbf{P}'_c)^\intercal = \mathbf{\Phi}_c(\mathbf{P}'_c)^\intercal$ and $\mathbf{\Phi}\mathbf{P}_c^\intercal = \mathbf{\Phi}_c \mathbf{P}_c^\intercal$. If the variance $\mathrm{Var}\left[\mathbf{\Phi}(\mathbf{P}'_c)^\intercal\right]$ were larger than $\mathrm{Var}\left[\mathbf{\Phi}\mathbf{P}_c^\intercal\right]$, then $\mathrm{Var}\left[\mathbf{\Phi}_c(\mathbf{P}'_c)^\intercal\right]$ would be larger than $\mathrm{Var}\left[\mathbf{\Phi}_c\mathbf{P}_c^\intercal\right]$, contradicting that PCA maximizes the variance. □

According to Props. 1 and 2, a projection $\mathbf{P} \in \mathbb{R}^{d' \times D}$ compatible with $\{\mathbf{w}, b\} \in \left\{\mathbb{R}^D, \mathbb{R}\right\}$ and that maximizes the retained variance is given by $\mathbf{P}^\intercal = (\mathbf{w}/\|\mathbf{w}\|, \mathbf{u}_{c,1}, \mathbf{u}_{c,2}, \ldots, \mathbf{u}_{c,d'-1})$, with $\mathbf{u}_{c,i}$ the eigenvector with the i-th largest eigenvalue of $\mathbf{\Phi}_c \mathbf{\Phi}_c^\intercal$ and $\mathbf{\Phi}_c = \mathbf{\Phi} - \mathbf{\Phi}\mathbf{p}_1\mathbf{p}_1^\intercal$. The overall proposed visualization procedure, for a NN with a single logistic/linear output unit, is shown in Alg. 1.

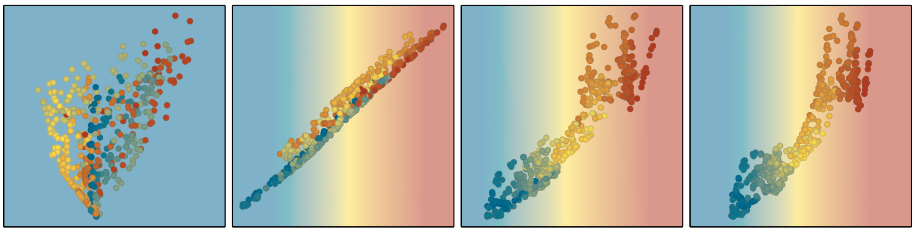
4 Experiments

This section shows how the proposed method works in two synthetic datasets and a real-world problem. The visualization method is implemented in Python, and the Neural Networks are built using Keras.

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2. Available from http://www.i6doc.com/en/.



(a) `moons`: Untrained. (b) `moons`: 30 epochs. (c) `moons`: 60 epochs. (d) `moons`: 90 epochs.



(e) `roll`: Untrained. (f) `roll`: 20 epochs. (g) `roll`: 40 epochs. (h) `roll`: 60 epochs.

Fig. 2: Evolution of the NN over the `moons` and `roll` datasets.

The first dataset, `moons` (Fig 1a), is a classification problem generated using the function `make_moons` of *Scikit-learn* with 250 points per class and a noise level $\sigma = 10^{-1}$. Figure 2 (top) shows the evolution of a FFNN with two hidden layers of 10 units each and ReLU activations. The colours of the points represent the real class, the background colour the prediction (which only depends on the x-axis) and the line the separating hyper-plane. These plots show how the NN iteratively untangles the problem until it is (almost) linearly separable.

The second example, `roll` (Fig. 1d), is generated with `make_swiss_roll` of *Scikit-learn*, with 250 patterns per class and without noise. As before, Fig. 2 (bottom) shows the evolution of a FFNN with the same architecture as above. Ideally, the colours of the background and the points should be the same. Again the NN unrolls the dataset gradually until arriving to an almost linear problem.

The last problem (`mnist`) is the MNIST dataset provided by Keras. Two classes are selected, corresponding to digits four and nine, and 500 patterns of each class are sampled. Three images of digit one are also included in the class of digit nine to study where the visualization method locates these outliers. The NN used is the Convolutional NN (CNN) described in the documentation of Keras for this dataset, and it is trained during 10 epochs. The resultant visualization is depicted in Fig. 3, together with the images corresponding to the patterns on the border of the cluster of each class. It is easy to see that the different regions of the space correspond to different ways of writing each digit. Moreover, the three outliers appear concentrated in the lower part of the cluster of digit nine (i.e., the second dimension is providing information in this sense). There is also a misclassified point, a four that clearly resembles a nine.

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2. Available from http://www.i6doc.com/en/.

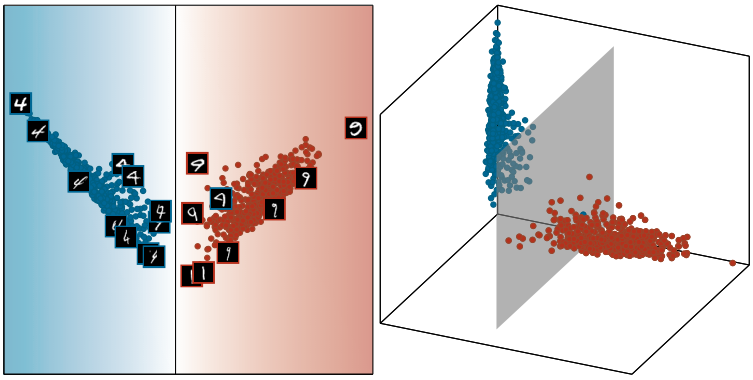


Fig. 3: Visualization of the CNN for `mnist` dataset in two and three dimensions.

5 Conclusions

Visualization is receiving great attention within the ongoing efforts to interpret Machine Learning models. In the case of Neural Networks (NNs), most work focuses on reconstructing the samples that most excite the network. A different approach has been proposed here, depicting the patterns after the transformation that takes place in the NN, in a way compatible with the linear model on top of the NN. This visualization requires reducing the dimensionality from D (the number of units in the last hidden layer) to two/three so that it can be plotted. This work has proved that the reduction compatible with the linear model that more variance retains is formed by a projection over the hyper-plane that defines the linear model plus a PCA projection over the orthogonal space. As shown experimentally, this approach can be useful for studying the evolution of the model with respect to the training epochs and to have a representation of the data compatible with the NN. It can also be applied to detect outliers, or to cluster the patterns according to their behaviour with respect to the NN.

As further work, this kind of approach can be applied to other Machine Learning models that include, either explicit or implicitly, a feature transformation plus a linear model.

References

[1] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. In *Deep Learning Workshop, ICML*, 2015.

[2] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV 2014*, pages 818–833, Cham, 2014. Springer International Publishing.

[3] L. M. Zintgraf, T. S. Cohen, T. Adel, and M. Welling. Visualizing deep neural network decisions: Prediction difference analysis. In *ICLR Conference Track Proceedings*, 2017.

[4] A. Schulz, F. Hinder, and B. Hammer. Deepview: Visualizing the behavior of deep neural networks in a part of the data space. *arXiv:1909.09154*, 2019.

[5] J. P. Cunningham and Z. Ghahramani. Linear dimensionality reduction: Survey, insights, and generalizations. *The Journal of Machine Learning Research*, 16(1):2859–2900, 2015.