MARCEL JUNIOR WANDJI
10/02/2021

Implementation of a simulation to explore a computational model of visual concept learning

1. Introduction

The goal of this simulation is to train a model with a given dataset and explore how the model learn to represent the data. The model is trained with an unsupervided learning algorithm and implemented in Matlab.

2. Model Structure

The model used in this simulation is the Deep Belief Network. It is a stack of the restricted Boltzman machine. The network architecture for this simulation is made up of tree hidden-layers: the first two layers are made up of one hundred neurons and the last is made up of 500 neurons. The number of maximum epoch is set to 30, the batch size to 300, the sparcity set to 1 to encourage the sparsity on the third layer. The complete setup of the network can find on the *deeptrain_CPU.m* file.

3. Dataset

The dataset used for this simulation is the eminist-digits which contains 280000 written digits. The dataset is divided into training set which contains 240000 images of written digits and testing set which contains 40000 images. In the figure below we can see some digits from the training set.

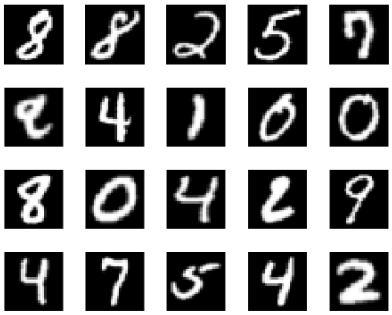

Figura 1. Data from the training set of emnist-digits dataset.

The dataset has been splitted into batches to improve the learning speed. The training set is splitted into eight hundrend batches, each containing three hundred images. The testing set on the other hand was splitted into two hundred batches with two hundred images each.

4. Training

The learning is done on a single layer of the boltzman machine, which means each layer is trained at the time. After training the first layer the weights are being freeze and the training moves to the next layer. This is done until all the anterior restricted boltzman machine are trained. At the first layer the given input is the data divided into batches and the input of the next layer is the output of the previous layer and they are trained for 30 epochs. The algoritm used to train a single layer (a single layer is seen as a *Restricted Boltzman machine*) is the contrastive divergence algoritm. This algoritm is done in two phases: the positive and the negative phase. In the positive phase the pattern is presented to the network, the activations of the hidden units are computed in a single step using the sigmoid function. Then we evaluated the correlation between the visible and the hidden layer and we sample the active units to which one is activated. The sampling result is used for the negative phase. In the negative phase we reconstruct the visible units starting from the hidden layer activations. After that we evaluate the correlation between the two layers in order to update the weights.

5. Observations

After the deep belief network was trained, we can plot the receptive fields (connection weights) of the three layers, to see how the features of digits are represented at each layer as shown on the figures below.

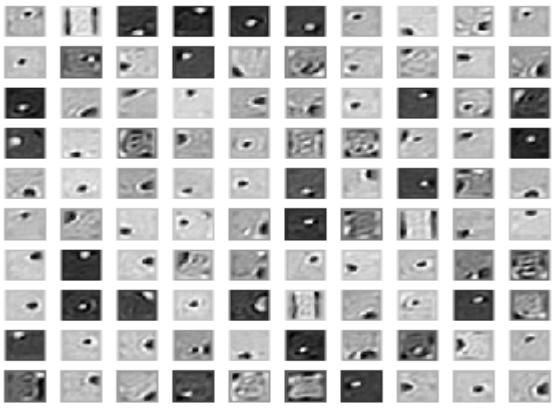

Figura 2: Receptive fields of the first layer.

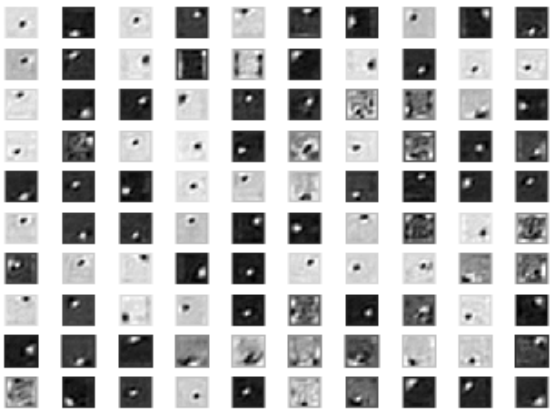

Figura 3: Receptive fields of the second layer.

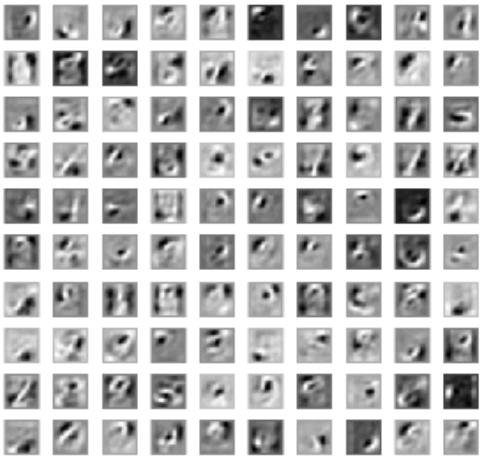

Figura 4: Receptive fields of the third layer.

Each image on the receptive field represents the weight of the connection between one hidden unit and the visible unit. Each unit respond to a particular feature present on an image, for example a line or a stroke in a particular position in the image. At the third layer we see that the network combine the features extracted in the layers below to contruct more complex features that may look like digits. The result of this learning in general is that the network extract features that can best describe the data.

To understand the features extracted at each level we perform the linear read-out at different level of the hierarchy by training supervise layers to map the internal representation of the deep belief network at different level of the hierarchy. The result of the linear read-out is given in the figure below.

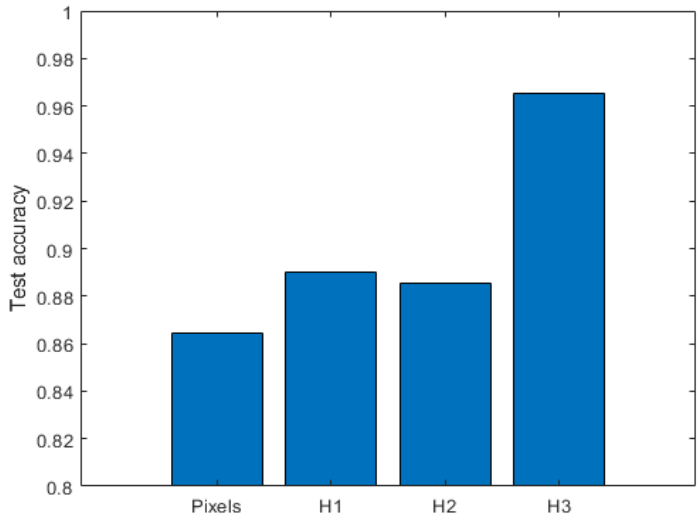

Figura 5: Classification test accuracy for each layer.

We can observe that given the raw image to the trained supervised layer (perceptron) it has a test accuracy of 86%. We futher observe that the performance of the first and the second layer does not differ so much but increases compare to the raw image because data given to the classifier at these levels is more significative. Using more complex features in the third

layer we observe that the classifier improove considerably the performance to classify digits. The higher the representation of the data the better classification.

To analyse the results on the testing dataset in detail we can observe the confusion matrix after the classification at the third layer on the figure below.

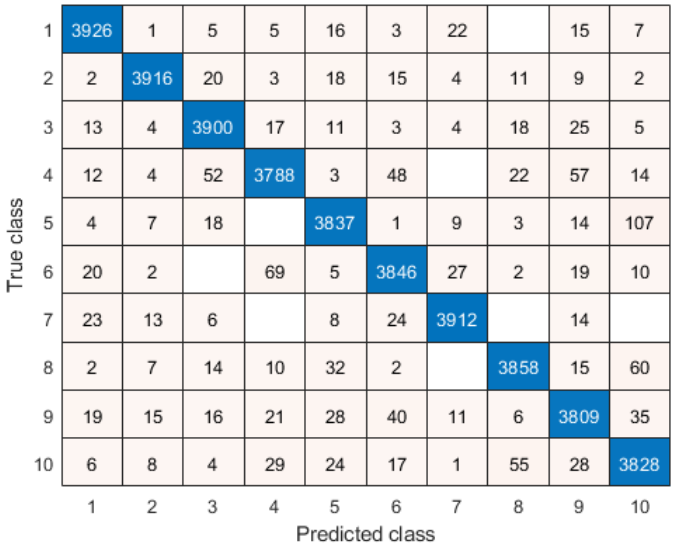

Figura 6: Confusion matrix after the classification at the third layer.

We observe that the model some times makes more confusion with the digits 4 and 9 (on the matrix is represented by the classes 5 and 10 respectively) by predicting 9 when the actual value is 4. The reason for this might be the similarities between their features which we notice observing Figura 1. We futher observe the error to classify properly the digit 3. The model predict 57 times the digit 3 to be 8. Even in this case the reason is that these two digits presents some similar features. We observe the same situation in predicting 55 times the digit 9 to be 7.

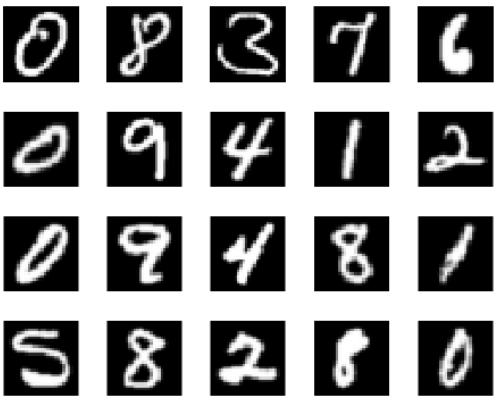

Figura 7: Testing data

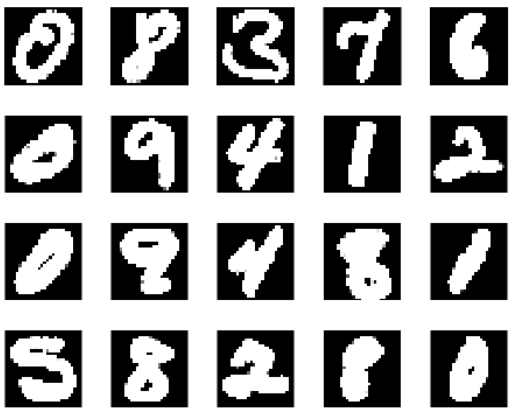

Figura 8: Noisy testing data

The figure 8 above show the test images containing some noise. The more the images are noisy the less is the accuracy of the model. Observing the confusion matrix (Figura 8) of the classifier at the third level, the classifier makes more errors predicting 9 when the actual image is 4. We also notice that more errors are made in other cases due to noise, for example predicting 7 when the actual value is 1. This must be due to the fact that the noise added to the test image is actually hidding some significant features that helps the model to contruct a proper representation.

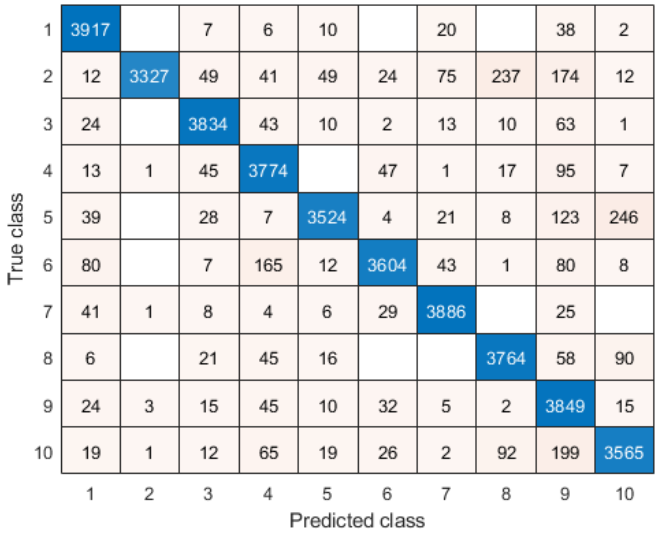

Figura 9: Confusion matrix with noisy test image.