Forming sparse local representations by anti-Hebbian learning using the MNIST dataset

Caroline Kellner

May 23, 2023

1 Introduction & Methods

The learning tasks proposed in Földiak (1990) use input that is notably smaller than the images of the MNIST dataset (LeCun et al., 2010). Additonally, due to being artificially created, the samples comprising the same label have less variability than the handwritten digits in MNIST. Given these differences, I am interested in finding out if the network structure proposed by Földiák is able to represent the MNIST dataset in a sparse coding manner.

1.1 Dataset

MNIST is a dataset of handwritten digits. It is composed of 70000 gray scale, 28x28 pixel images and their according labels. The dataset is divided into a training and a test set of sizes 60000 and 10000. I use these datasets respectively to train the network and test it's representation.

1.2 Network

The network comprises one layer of 10 Hebbian units. They receive input from 784 input units which are connected to the perceptrons using a dense layer. The Hebbian units have inhibitory connections to each other. A visualization of the network can be seen in Fig. 1. During training the hyper-parameters of the network are set in the following way: $\alpha = 0.01, \beta = 0.001, \gamma = 0.01, \lambda = 10, p = 0.1$. The Hebbian connections are initialized randomly, as are the thresholds of the neurons. The anti-Hebbian connections are initialized as 0. Before training the weight matrices the thresholds were pre-trained by running the network with $\alpha = \beta = 0, \gamma = 0.1$ with an input of 1000 randomly selected images.

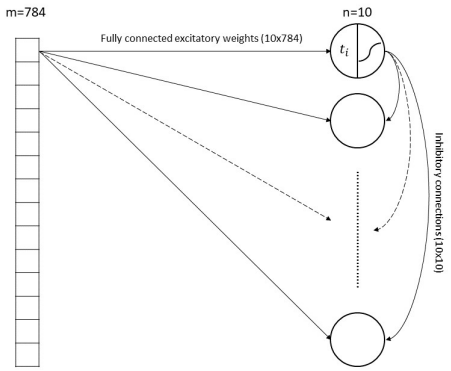

Figure 1: Structure of the Hebbian/Anti-Hebbian network

2 Results

After training the network for 5 epochs the weight matrix shows features of the input (Fig. 2). Interestingly, the receptive fields do not code for every single digit in the same way but rather show redundancies and missing digits. For example, the fields 0, 4 and 7 show a form resembling a zero, while no field can clearly be identified as coding for a 1 or 7.

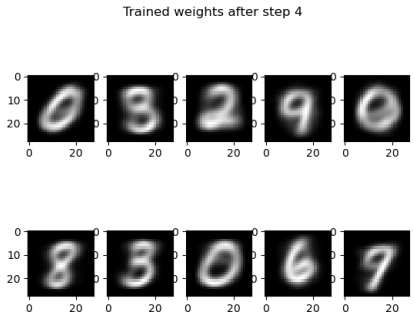

Figure 2: Receptive fields of the units after 5 training epochs

Fig. 3 shows the mean output vectors, as well as example output vectors of the network, encoding a single digit. Most of the digits show preferred output neurons that activate more often in response to the digit than other neurons in the network. The classification does not work perfectly though, as there is still some variation in the activation.

The example output vectors show that only a few neurons are active per digit at the same time. An interesting example is the digit 1, as it is the digit with the most occurrences in the MNIST dataset and simultaneously coded with only inactive neurons in this network. From the mean vector it is visible that this is not a mere coincidence but that input with the label "1" shows a generally low probability for active neurons. This is similar to Földiak (1990) where more frequent letters are coded using less active bits.

a)

```
Mean vector for digit 0:
[0.3622449  0.09285714 0.08163265 0.05306122 0.29489796 0.00816327
 0.03571429 0.62244898 0.07040816 0.00816327]
Mean vector for digit 1:
[0.         0.00352423 0.00881057 0.         0.         0.07577093
 0.00088106 0.         0.00881057 0.00176211]
Mean vector for digit 2:
[0.04360465 0.16472868 0.65503876 0.05910853 0.11918605 0.07364341
 0.02325581 0.01937984 0.0620155  0.00484496]
Mean vector for digit 3:
[0.01980198 0.3960396  0.08118812 0.04257426 0.0049505  0.08217822
 0.46039604 0.10891089 0.04059406 0.00990099]
Mean vector for digit 4:
[0.02851324 0.02342159 0.01120163 0.34521385 0.21894094 0.00814664
 0.00101833 0.         0.09775967 0.12525458]
Mean vector for digit 5:
[0.17825112 0.2544843  0.00784753 0.02690583 0.02242152 0.10201794
 0.16591928 0.16143498 0.03699552 0.01121076]
Mean vector for digit 6:
[0.13361169 0.02818372 0.11586639 0.13048017 0.29436326 0.00313152
 0.02505219 0.00730689 0.58350731 0.         ]
Mean vector for digit 7:
[0.02042802 0.01459144 0.0233463  0.13132296 0.06031128 0.03404669
 0.         0.         0.00583658 0.33657588]
Mean vector for digit 8:
[0.04825462 0.29876797 0.04620123 0.12217659 0.04517454 0.61601643
 0.15297741 0.02772074 0.15913758 0.07392197]
Mean vector for digit 9:
[0.03666997 0.04063429 0.00693756 0.42913776 0.14370664 0.05649158
 0.01783944 0.00693756 0.06342914 0.26263627]
```

b)

```
0 ::: [[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]]
1 ::: [[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
2 ::: [[0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]]
3 ::: [[0. 1. 0. 1. 0. 0. 0. 0. 0. 0.]]
4 ::: [[0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]]
5 ::: [[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
6 ::: [[1. 0. 0. 0. 0. 0. 0. 0. 1. 0.]]
7 ::: [[0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]]
8 ::: [[0. 0. 0. 1. 0. 1. 0. 0. 1. 0.]]
9 ::: [[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

Figure 3: a) Mean output vectors for single digits. b) Example output vectors for single digits

3 Discussion & Outlook

As can be seen from the receptive fields the network is able to recover features from the dataset. The example vectors consisting mainly of 0 with rare 1s, show that the decorrelation between the active neurons work and that the inhibitory connections suppress output from other neurons. There are however also vectors with more than one active neuron, which means that it is still possible to code for input using multiple features. From the mean vectors we can see that the most used neurons are distributed between the different labels. The digits share some of the representational units though, which means that the network might be able to represent reoccurring features of the input such as angles, straight lines or closed off areas.

One of the limitations of this version of the network is that the 10 output neurons are theoretically able to code for each one of the digit labels in a distributed manner. So while the decorrelation of the neurons works well, I find it difficult to make clear claims about actual sparse coding and feature combinations. Theoretically 4 representational units would be enough to represent all 10 digits using a binary coding. Going forward it could be interesting to see if the network is able to learn the encoding for the MNIST dataset using less output neurons.

On a different note, for my master's thesis I am working on clustering different musical genres represented by multidimensional feature vectors to organize them in a two-dimensional space. Currently I mainly use different implementations of Principal Component Analysis (PCA) for this. One limitation of this is the linear character of the classical PCA. Földiák's algorithm has been described as being able to pick up the principal components of the input dataset, so I would be interested in using the general algorithm and tweaking it in a way to find the two main components of the musical feature vectors.

References

Földiak, P. (1990). Forming sparse representations by local anti-hebbian learning. *Biological cybernetics*, *64*(2), 165–170.

LeCun, Y., Cortes, C., & Burges, C. (2010). Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, *2*.