Unsupervised and reinforcement learning in neural networks
Project 1
Kohonen maps on hand-written digits

Joachim Muth and Sebastien Speierer

November 17, 2015

Project description

For this project, our task is to understand the Kohonen algorithm and apply it to a data set of hand-written digits. A Kohonen map is an artificial neural network trained using unsupervised learning to produce a low-dimensional representation of the input space. For this exercise, our inputs consists of vectorized 28x28 images of digits so it has a dimension of 784. Thus, the Kohonen map will be very useful allowing us to look at this data on a two dimensional map.

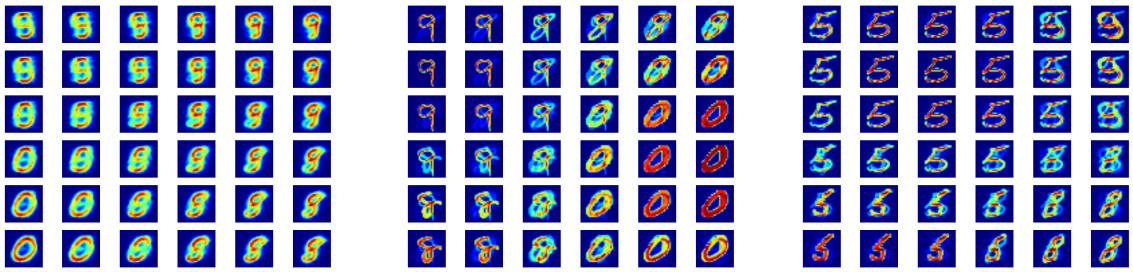
Through this report, our subset of digits is composed of the labels

$$[0, 5, 8, 9]$$

1 Choice of learning rate and convergence criteria

Learning rate

Here are different examples of clustering with different values of η:



(a) $\eta = 0.1$ (b) $\eta = 1$ (c) $\eta = 1$

Figure 1: Clustering with different learning rate

On the Figure 1 (a), the prototypes look like a mix of all the different inputs. This is due to a too small η. Prototypes don't have time to specialize since they are learning too slowly.

On Figure 1 (b), the prototypes are really specialized thanks to the high learning rate ($\eta = 1$). Notice that the data with label '5' don't show up and this is due to the order in which data are presented. For this figure, the 5 last data inputs given to the algorithm were labeled: $[..., 8, 8, 8, 0, 9]$
There is no '5' in these last data inputs and since the learning rate is really high, the prototypes focused on these labels really quickly.

On Figure 1 (c) we can observe the opposite situation, where we added some '5' labeled data at the end of the sequence. In this case, the 6 last data given to the algorithm were labeled: $[..., 9, 8, 5, 5, 5]$

By these observations, we understand why extreme values of η won't yield in accurate cluster of the overall data inputs. We have to find a good trade-off between these extremes.

We want the magnitude of the change decreases with time, and we also want our algorithm to learn pretty quickly at the beginning so our prototypes can quickly jump to the center of a cluster. One way of doing it is to set a high initial $\eta_0 = 0.7$ and decrease it over the iterations. Thus we don't waste everything we've learned so far in the case where a sequence of inputs with the same label occurs. This yields a slower convergence but more accurate predictions.

$$\eta_0 = 0.7, \eta_n = \max(0.9999\eta_{n-1}, 0.1)$$

Convergence criteria

We define the convergence of our algorithm when the assignments no longer change or when we've done more than $tmax = 20000$ iterations.

At the Nth iteration we can compute the error term as

$$E_n = \sum_{n=0}^{N} |(C_n - C_{n-1})|^2$$

with C the set of centers at the nth iteration.

On figure 2 (a) we can see these error terms decay over the iteration. Because of the high variance of theses values, we need to take the mean over the last 500 error terms which is defined by

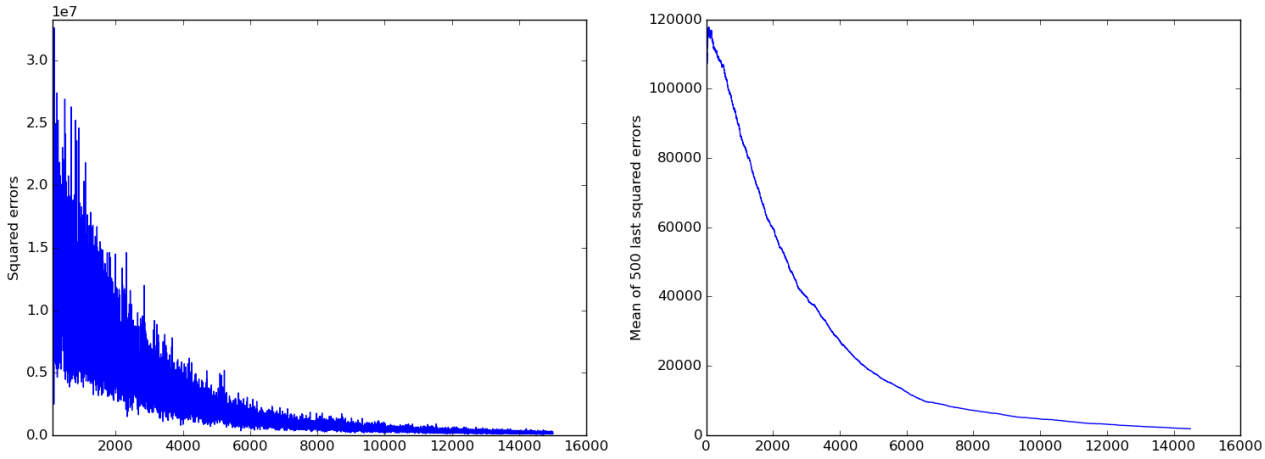
$$ME_n = \frac{1}{500} \sum_{i=0}^{500} E_{n-i}$$

Figure 2 (b) shows the variation of this mean error over the iteration and we see that we've definitely got rid of the variance there.

We now just have to fix a convergence criteria ϵ such that whenever $ME < \epsilon$ we stop the algorithm.

In our case we set

$$\epsilon = 1000$$



(a) E_n over the iterations

(b) ME_n over the iterations

Figure 2: Error terms used as convergence criteria

2 Method for assigning digit

After creating a self-organized map with the Kohonen method, each adjusted center need to be assigned to a numerical value.

In order to do that, we will reverse the process used to assigned data to a center. Using a root mean square metric, we find for each center the nearest data from it. Once we got the index of this particular data, we search into the label database which digit correspond to it and print it.

We finally obtain a matrix of values that correspond to each center in the Kohonen map. Obviously, due to the inner implementation of a self-organized map, not all the centers are well assigned to a digit. Some of them are influenced by multiple cluster and mixes some digit. For them, the assigned digit will be the nearest that algorithm has found.

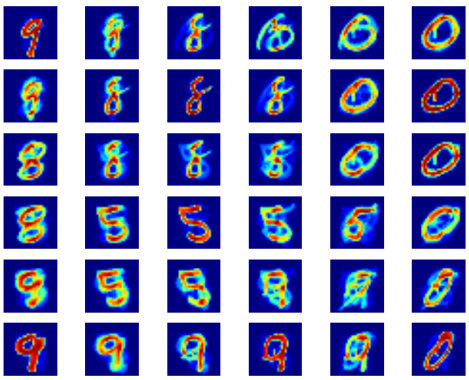
The following code implements the described method:

```
# this selects all data vectors that corresponds to one of the four digits
data = data[np.logical_or.reduce([labels==x for x in targetdigits]),:]
# filter the label
labels = labels[np.logical_or.reduce([labels==x for x in targetdigits])]

...

# Find the digit assigned to each center
index = 0;
digits = []
for i in range(0, size_k**2):
    index = np.argmin(np.sum((data[:] - centers[i, :])**2,1))
    digits.append(labels[index])

print np.resize(digits, (size_k, size_k))
```



(a) Kohonen map

```
[[ 9.  9.  8.  0.  0.  0.]
 [ 9.  8.  8.  8.  0.  0.]
 [ 8.  8.  8.  8.  0.  0.]
 [ 8.  5.  5.  5.  5.  0.]
 [ 9.  5.  5.  9.  9.  0.]
 [ 9.  9.  9.  9.  9.  0.]]
```

(b) Digit assigned to each center

Figure 3: Digit labeling of each center of Kohonen map

3 Width of the neighborhood function and network size

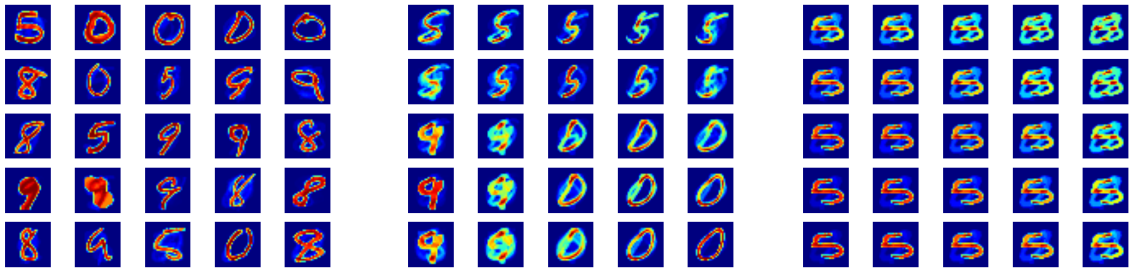
3.1 Neighborhood width

Theses parameter influence the algorithm in the following ways:

- When we assign a new data input to the nearest center and update its position, the **neighborhood function** update also the neighbors of this center decreasingly, according with the width of the Gaussian function it uses. Thus, a small width σ will result in few neighbor influences and an algorithm similar to competitive learning algorithm (where some center corner the data from the very principle and other center just dies). Inversely, a big σ will influence every center for each data input, and all of them will finish representing the same thing (the center of the points).

- A useful propriety of Kohonen map is that the neighborhood function prevent the algorithm from overfitting. It means that even if it runs over a great number of variables (centers), theses variables will still group themselves around clusters. Obviously the map need to be large enough to represent each type of data. A good approximation could be a number of centers equal to ten times the number of expected cluster. $\#centers \approx 10 * \#clusters$

Both of theses two parameters depend from each other, as the sigma σ represent a Gaussian distribution over the map, its size must depend of the size of the map and vice-versa.

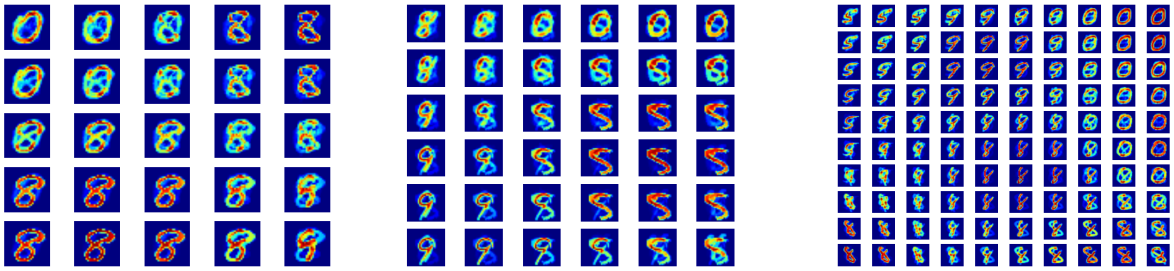


(a) $\sigma = 0.4$
Each center is nearly independent of the others and there is no cluster construction

(b) $\sigma = 1.2$
For a square width of 5, such a sigma is a good option. Indeed all kind of input are represented and regrouped.

(c) $\sigma = 5$
Each center being influenced by every data, a too high sigma will result in a map containing all the same elements.

Figure 4: Clustering with different width σ of neighborhood function over a map of constant width $w = 2$



(a) $width = 4$
Not enough centers to represent the variety of cluster. Possible to guess zero at the top-left position

(b) $width = 6$
Good design with representation of every cluster.

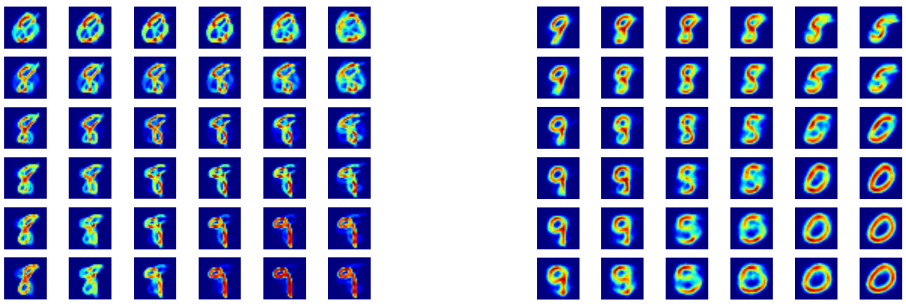
(c) $width = 10$
Also good design with clear representation of each cluster easy to differentiate and well grouped.

Figure 5: Clustering with different map width with constant sigma $\sigma = 2$

4 Variation of the width of the neighborhood function over time

We can improve the algorithm by beginning with a high sigma σ and decreasing it over the time. This allow the algorithm to quickly move all the centers in the area where are the data and let them gradually more independence to form different cluster. Here is provided the implementation and the result of such an improvement

```
def som_step(centers,data,neighbor,eta,sigma):
    ...
    # decrease the sigma
    sigma[0] = max(0.9995 * sigma[0], 1.0)
```



(a) $\sigma = 2$ no time dependence

(b) $\sigma = 10$, decaying over time 0.0001% each step

Figure 6: Variation of sigma over time

5 Results

On Figure 7 you can finally observe our final Kohonen map of inputs with labels $[0, 5, 8, 9]$.

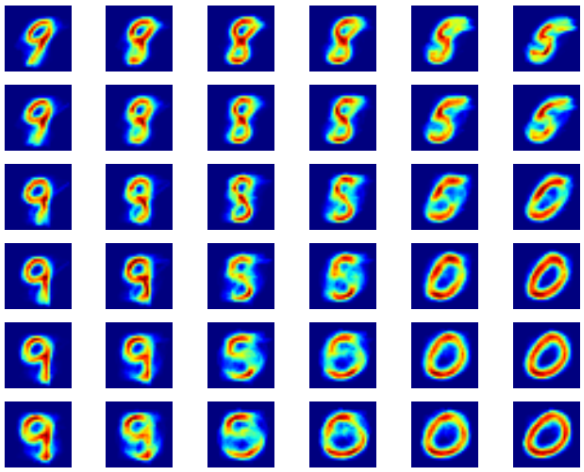


Figure 7: Final Kohonen map computed with 15000 iteration

On this map we can clearly distinguish the different labels on different clusters. Also, we still have undecided blurry prototypes in between these clusters but this isn't an issue since we only care about the best match to classify our data inputs. Also, with the help of some image processing filter, like a local-normalization filter which attenuates the non uniformity of the background, we could get even better result.

(a) label '0'

(b) label '0'

(c) label '8'

(d) label '9'

Figure 8: Best prototypes for the different labels

Finally, it is good to have different prototypes representation for each label since each digit can be represented in different ways (look at the different '9's on the first column on Figure 7 for instance)