EPFL

Unsupervised and Reinforcement Learning in Neural Networks

Kohonen maps on hand written digits

Lecturer: Marc-Oliver Gewaltig

Teacher Assistant: Dane Corneil, Berat Denizdurduran

Students: Chiara Gastaldi and Gonçalo Cardoso Rodrigues Bonifácio Vítor

- 30-11-2014 -

Choice of learning rates and convergence criterion

We start by considering a Kohonen network of 6×6 neurons arranged on a square grid with unit distance. In the algorithm we use a Gaussian neighborhood function Λ, with constant standard deviation. There are $m = 2000$ data samples specific to our data set (corresponding to 1, 4, 8 and 9 using the name 'Chiara Gastaldi'), k is the number of prototypes (we start with $6 * 6 = 36$ prototypes) and $n = 784$ is the dimensionality of the data samples and prototypes. The Kohonen algorithm is described by the following steps:

1. The k prototypes $w_1, w_{2,...,}w_k$ are randomly initialized (n-dimensional weight vectors).
2. The input pattern x^μ is picked at random from our data set.
3. The winning prototype w_j is determined by choosing the prototype that has the smallest distance to the input pattern:
$$\left|w_j - x^\mu\right| \leq |w_i - x^\mu| \quad \forall i = 1, ..., k.$$
4. All the prototypes are updated using the neighborhood function and the online update rule:
$$w_i \rightarrow \eta\Lambda(i, k)(x^\mu - w_i), \quad \forall i = 1, ..., k.$$
5. The size of the neighborhood function is decreased (skipped for now, we consider it constant).
6. Go back to point 1 and iterate until stabilization.

Convergence criteria

For the convergence criteria we tried the following convergence metric: $all(abs(centers - new\_centers) < threshold)$, which would mean that no prototype (vector) component should differ from the previous value more than the threshold. This rule was associated with a stopping criteria for a certain t_{max} to ensure the algorithm would stop. We tried setting the threshold to different values but the algorithm didn't converge so we decided to analyze the value of our convergence metrics along time and we realized that it didn't significantly change over time.

We also tried a different metric: $max(sum((centers - new\_centers).^2, 2)) < threshold$, which means that the distance moved by each prototype should be inferior to the threshold. Again we obtained the same result as before (in both cases using very small learning rates from .5 to .05) and thus we decided to analyze our results visually and based on different values of t_{max}. For this parameter we set the value to 5000, 10000, 20000 and 50000. Since we didn't observe any significant differences between the tests made using the larger two values, we decided to drop the last value, as it was computationally very expensive.

Since the algorithm should converge when a suitable (small) learning rate is chosen we can only infer that the results obtained are linked to not decreasing the neighborhood function.

The learning rate

The learning rate is the coefficient that tells how much the prototype should be moved in the direction of the new assigned data point, and thus influences the speed of the algorithm.

The instruction sheet suggests to choose $\sigma = 3$ and look for a constant η such that the algorithm would converge. However, we tried many values of η and t_{max} keeping $\sigma = 3$ but the convergence was never achieved, even with very small η combined with very long t_{max}.

For a fixed sigma, we varied η and t_{max} to see their impact on the prototypes. Then, for a given fixed η and plotting the prototypes for different values of t_{max}, we observed that the prototypes tend to get blurred as t_{max} is increased and thus the algorithm does not converge and yields better results for smaller values of t_{max}.

The neighborhood function

On the other hand, the value of σ determines the impact of the classification of the current data point in the neighborhood of the winning prototype. This parameter is used to distort the tessellation of prototypes so that regions with higher densities of data samples will be covered with a higher density of prototypes.

Holding η constant and equal to 0.05, $t_{max} = 20000$, and for σ equal to 1, 3 and 5, we observe that as σ is increased, there is less variety in the cluster labels (more prototypes are assigned to the same label). This can be easily interpreted as a too big distortion of the grid of prototypes, i.e. when a data sample is clustered to a prototype, the other prototypes are too attracted to this winning prototype, hence leaving large regions of the n-dimensional space uncovered. Therefore, for bigger values of σ we should decrease t_{max} so that we avoid the blur effect (see Fig. 2 and Fig. 3). Furthermore, the variations caused by the σ for several stopping criteria were tried and similar results were observed by visual analysis.

On the other hand it was possible for the algorithm to converge (on the proposed metrics) either by (1) decreasing the value of σ holding η constant or (2) by doing the opposite, but since it was not mentioned (2), we only include the results for (1) more ahead.

Visualization and description of the learnt prototypes

Classification of the prototypes

For the classification of the learnt prototypes we tried two different methods. The first method consisted on storing all the indexes of the data samples assigned to each prototype on a given run of the algorithm and then performing the histogram of the associated labels to that prototype. The most frequent label (the mode of the histogram) would be assigned as the label of the prototype. The second method consisted on assigning the label of the closest data sample to the prototype. As the n-dimensional space has a strong visual interpretation this method

allows for less misclassification of the prototypes than the previous one and hence was used to classify the prototypes in all our figures (indicated on the title).

The prototypes obtained for the choices of parameters described above are visualized in Figs. 1, 2 and 3. On the first, the prototypes are more clear as numbers can be perceived. On the second and third figures, the prototypes tend to get more indistinguishable and look very similar to each other. Again, since σ was kept constant, the results obtained are understandable.

The size of the network

In this paragraph we investigate if the optimal standard deviation of the neighborhood function σ varies with the network size. To do that we consider three different network sizes (6×6, 7×7 and 8×8) and three values of σ (1, 3 and 5).

From the analysis of Fig. 4, 5 and 6 we notice that in all cases $\sigma = 1$ provides good results, but also that as we increase the network size we get better result for bigger σ. We can explain it considering that the parameter σ impacts the region of influence of the winning prototype. For a small network we expect smaller optimal σ because the relative influence of the winning prototype on the network is bigger. And thus a value of *sigma* that is too large will impact too many prototypes, bringing them closer to the winning prototype. As we increase the network size we can expect that, gradually, the optimal σ will increase as well: in fact we want each each winning prototype to influence a bigger space area, so that we might have denser region of prototypes where the density of data samples is also higher, while not leaving other regions of the n-dimensional space uncovered (which can happen if the network size is small and the neighborhood is large). For graphical reasons we limited the network size to 8. Fig. 4, 5 and 6 are compliant with our analysis, even if a network size of 8×8 neurons is still not big enough to state that the optimal width of the neighborhood has increased.

Variation of the neighborhood function over time

However better results can be obtained if σ is decreased at each iteration. Trying different values of parameters we managed to get a result in which the prototypes are generally very acceptable and all the corresponding labels match (see Fig. 7). For the decay of sigma we used a logarithmic rule: $\sigma = \sigma_0 / log(t+1)$, and a value of $\sigma_0 = 4$. The results corresponding to this rule are shown in Fig. 7. We also tried $\sigma = \sigma_0 / t$ and $\sigma = \sigma_0 / \sqrt{(t)}$, but with both rules the algorithm converges too fast and not all data samples are successively clustered.

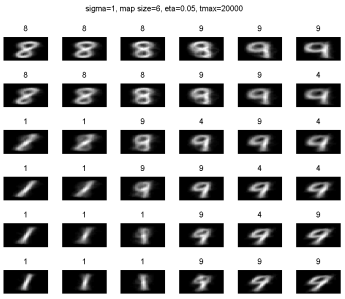


Figure 1: $\sigma = 1$.

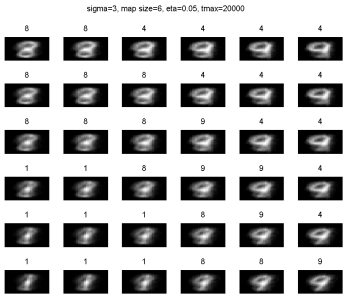


Figure 2: $\sigma = 3$.

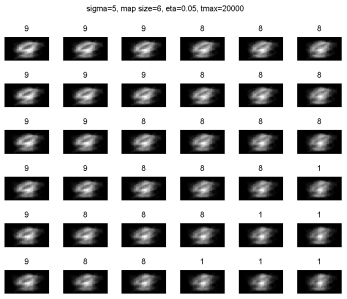
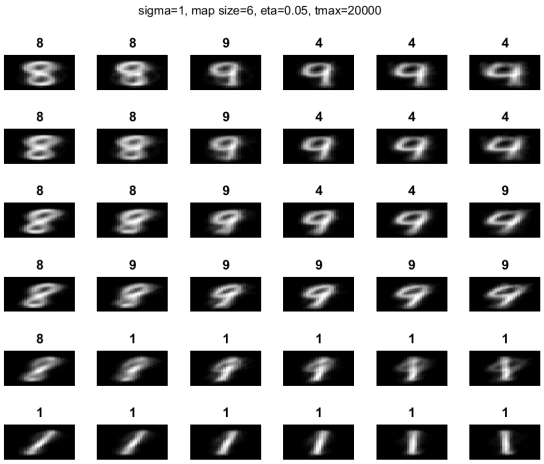
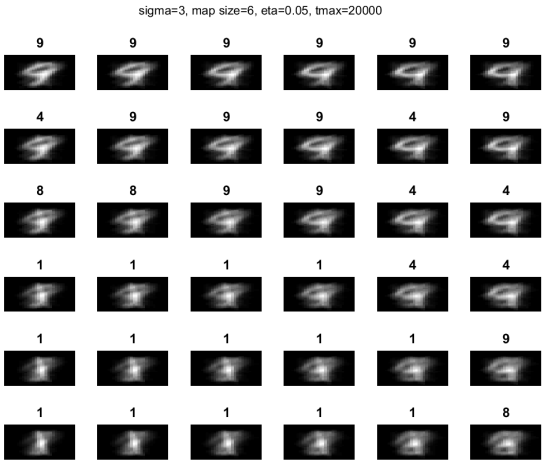


Figure 3: $\sigma = 5$.





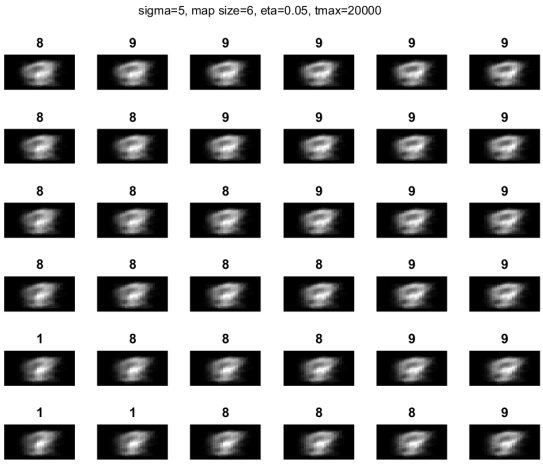


Figure 4: Results of the exploration of different network sizes and widths of the neighborhood function for a Kohonen network of 6x6 neurons.

Figure 5: Results of the exploration of different network sizes and widths of the neighborhood function for a Kohonen network of 7x7 neurons.

Figure 6: Results of the exploration of different network sizes and widths of the neighborhood function for a Kohonen network of 8x8 neurons.

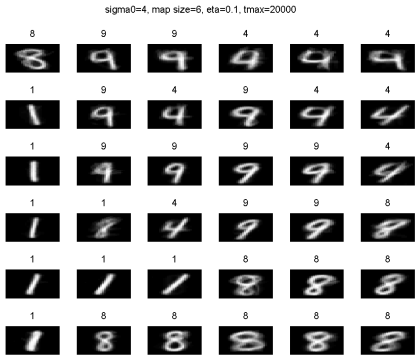


Figure 7: Varying neighborhood function over time using logarithmic rule.