ECOLE POLYTECHNIQUE FEDERALE DE LAUSANNE

# Unsupervised and reinforcement learning in neural networks
# Project 1

Flavio Martinelli and Tom Mingasson

May 4, 2018

## 1 THEORETICAL QUESTIONS

- There are only two layers in a Kohononen map connected by weights. The input layer of $N$ neurons receives the signal $x^\mu = [x_1^\mu..x_i^\mu..x_N^\mu]$ for the pattern $\mu$. The output layer of size $K$ computes for each neuron: $y^k = \sum_{i=1}^{N} w_{ik} * x_i^\mu$. Here for example $N = 28*28$ and $K = 36$.

- The four steps of the self-organizing map implemented in the Kohonen map defined as before are:

  1. *Initialization*: random initialization of the weights $w_{ik}$ for $i = 1..N$ and $k = 1..K$.
  2. *Competition*: for an input $x^\mu$ the neuron $k^*$ with the smallest distance $|w_k - x^\mu|$ is considered as the winner.
  3. *Cooperation*: A neighborhood around $k^*$ defined by a function $\Lambda$ is considered.
  4. *Adaptation*: All the weights $w_{ik}$ are updated using the on-line rule:
     $\Delta w_k = \eta * \Lambda(k^*, k) * (|w_k - x^\mu|)$ where $\eta$ is the learning rate.

- Through a Kohonen map an input signal $x^\mu$ from a space of dimension $N$ is assigned to a cluster $k$ hence mapped onto a neuronal space of dimension $K$.

- A parallel can be made between a Kohonen map and the topographic maps observed in the brain where input sensory signals (auditory or visual) are mapped onto a cortical space where input signals with similar features (frequency or orientation) excite neurons close to each other. Similarly in a Kohonen map input with similar features are assigned to clusters close to each other in the output space.

- Using Oja's rule which is a modification of the standard Hebbian's rule the weight update: $\frac{\mathrm{d}w_{ik}}{\mathrm{d}t} = \eta * y_k(x_i^\mu - w_{ik} * y_k)$ can be written as $\frac{\mathrm{d}w_{ik}}{\mathrm{d}t} = \eta * (x_i^\mu - w_{ik})$ when $k$ is the winning neuron since $y_k = y_k^2 = 1$, which is the Kohonen online learning rule.

## 2 Simulation of a Kohononen maps on hand-written digits

1. A reliable convergence measure has been proven hard to find, at first we monitored the value of the update of the centers and checked if consecutive updates presented a delta lower than a set threshold. This measure resulted unreliable and did not describe the evolution of the network, mostly because already learnt centers that resulted winner for certain samples did not contribute to an high shift in the centers positions, the resulting convergence measure was highly oscillating and difficult to tell when it settled.

   We opted for a measure of mean distances $m^i$ at iteration step $i$ between the training samples and the winning centers:

   $$m^i = \frac{1}{S}\sum_j^S \min_{n\in K^2} \left\| c_n^k - s_j \right\|$$

   where $S$ is the total number of samples in the training set ($S = 2000$), $c_i^k$ is the center of neuron $i$ computed at iteration step $k$, and $s_j$ is the $j_{th}$ sample in the set.

   This convergence measure better estimate how much the samples are, on average, close to the centers of the map. The goal of the Kohonen network is to learn to represent the input space in a lower dimensional space, by performing dimensionality reduction. The samples we have in the input space, which is 784dimensional, cover only a small subset of the total high dimensional space, therefore the actual subset can be represented by a limited number of points. The convergence measure described above will tend to a minimum as the number of neurons increase. Obviously incrasing too much the size of the network can be interpreted as overfitting, lost of generalization, this is why it is important to not increase the number of neurons indefinetly even if the performance will seem to get better and better.

   With this measure every network will converge to a different value of $m$ depending on how many neurons are present. To check when the network 'stops' learning we decided to look at the variation of the moving average of $m$ (with window size depending on the frequency we compute the $m$ value) along the iterations, when this value stops decreasing, meaning $m^{i+1} - m^i > 0$,we say the network has converged. This stopping criteria involves a trade off in the window size of the moving average, it must be enough large to account for oscillation in the values of $m$ that can bring the variation to a positive value, and enough small to check for the convergence in a computationally reasonable amount of iterations.

   The learning rate $\eta$ has been chosen reasonably small to get to convergence in a feasible amount of time and not too high in order to not overshoot the optimal points at the final stages of the learning algorithm, $\eta = 0.1$. It should be noted that to provide a fine

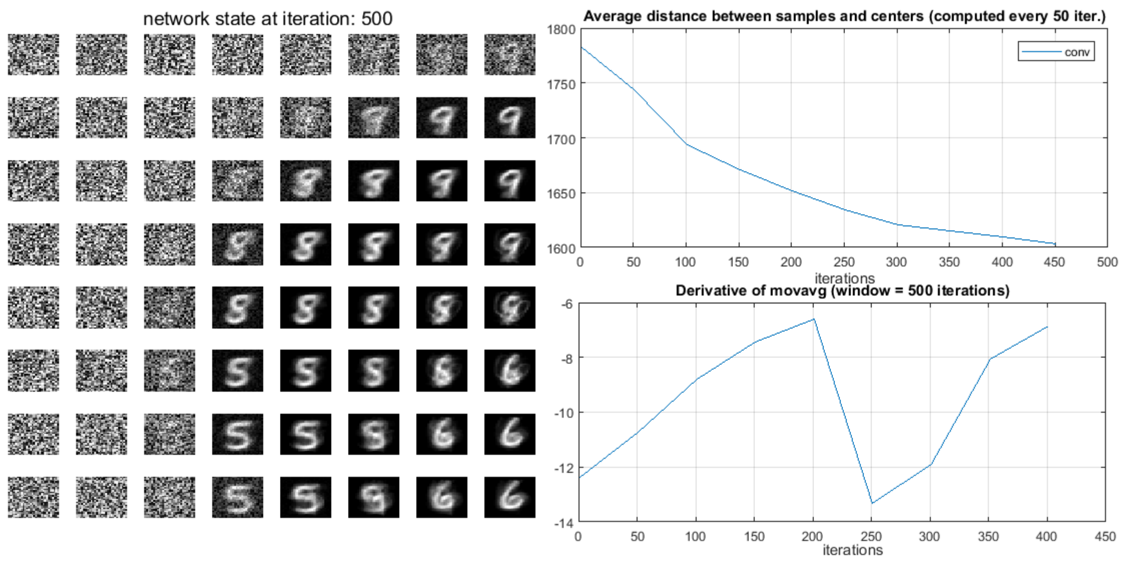


Figure 2.1: On the left: the state of the network centers at the $500_{th}$ iteration, on the right the measure of $m$ decreasing and the derivative of the filtered m, always negative.

tuning towards the end of the process the learning rate should be decreased as the number of current iteration increases.

The network shown here is composed of a unitary grid layer of $8x8$ neurons, $K = 8$, with gaussian neighboring function $\sigma = 1$. Here's an example of training of this map: fig.2.1, 2.2, 2.3.

2. In order to assign one digit to one prototype the euclidean distances to all the prototypes are computed then the winner cneter is the one for which the distance is minimal i.e the closest to the digit. In this way, after having trained a network using a dataset, it is then possible to assign all the digits in the dataset to one of the prototype. Finally it is possible to count in each prototype how many digits within each class have been assigned to it. This is possible since we have the labels of the digits.

   Figure 4(b) shows a final Kohonen map of size 6*6 after having been trained on the digits of the whole dataset whose labels are '9', '5', '6, or '8'. Figure 4(a) shows the labels distribution within each of the 36 prototypes. Figure 4(c) shows for each prototype the label with the highest count. In some prototypes (for example those in the first column) the distribution is strongly shifted to one of the label ('9' here). This makes sense since looking at the prototype shapes they are very close to a '9'. For other prototype like the one at (1st row, 2nd column) which is assumed to represent the label '5' the distribution tends to be much more uniform meaning that digits with different labels may have similar features in term of shape.

3. In order to assess the optimal values for $K$ and $\sigma$ an error function is defined and a K-fold cross-validation is implemented. To assess how well a given Kohonen map ex-

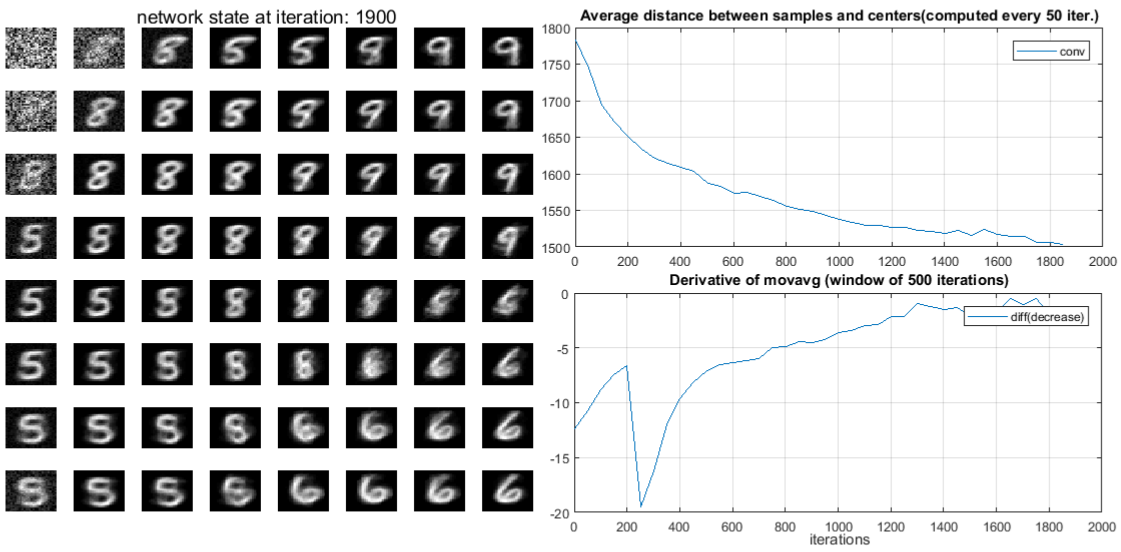


Figure 2.2: On the left: the state of the network centers at the $1900_{th}$ iteration, on the right the measure of $m$ decreasing and approaching a plateau and the derivative of the filtered m, always negative.

plains a data set of $N$ digits the error function computes the sum of the $N$ distances between the digits and their assigned prototype in the map. Then one can see if the map explains well the diversity of digits in the data set.

A cross-validation with 5-folds is implemented to assess the error of Kohonen maps trained with different values for $K$ and $\sigma$. An input dataset is split into 5 subsets. At each cross-validation step a Kohonen map is trained on the 4 subsets then the error of the obtained network is computed on the resting testing subset. A grid of 5*5 values of $K$ and $\sigma$ is investigated: $K$ takes values in $[6,7,8,9,10]$ and $\sigma$ in $[1,2,3,4,5]$.

Figure 2.5 shows the results in the parameter space. It can be observed that the optimal values are $K^{opt} = 10$ and $\sigma^{opt} = 1$. Note that for all the network sizes investigated the optimal neighborhood was always the smallest possible ($\sigma = 5$). This means that the best representation of the digits is obtained for large network using a small neighborhood for the training. Indeed when the network is too small there is not enough prototypes in the network to account for the variation of the digit shapes within the dataset. Also while learning with a large neighborhood, an input digit fed to the network spread during training result in an update of too many weights including those which have learned different labels hence avoiding clustering.

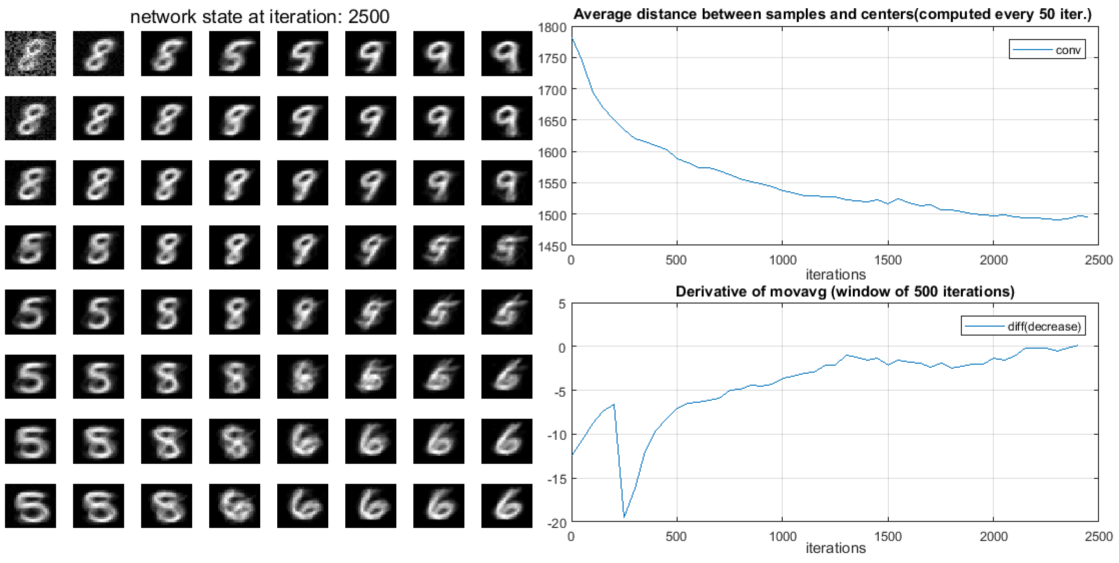
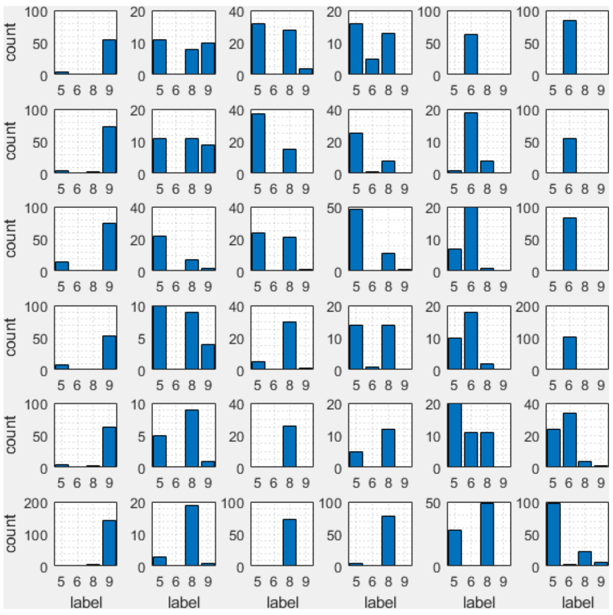
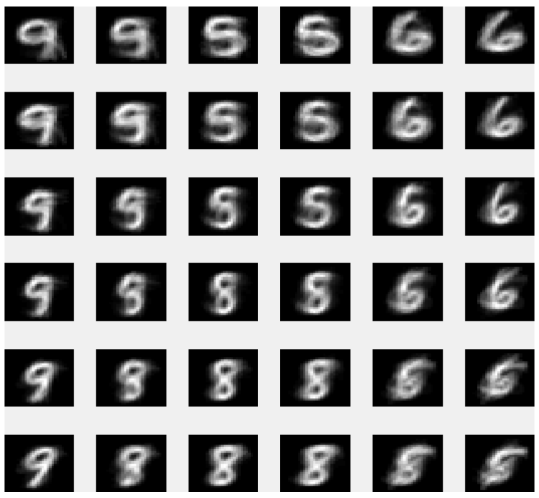


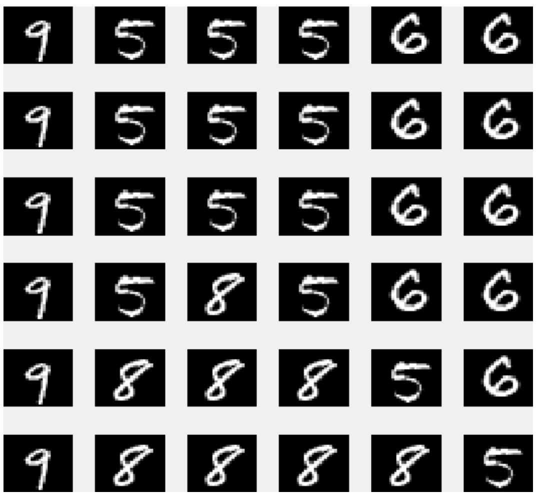
Figure 2.3: On the left: the state of the network centers at the $2500_{th}$ iteration, on the right the measure of $m$ approached the plateau and the derivative of the filtered m reached zero .



(a) Counts



(b) Trained Network



(c) Learned Labels

Figure 2.4: Assignement of digits to each prototype of a trained network (b) and visualization of the count in the prototypes with respect to their label (a). Best represent in each prototype in (c).

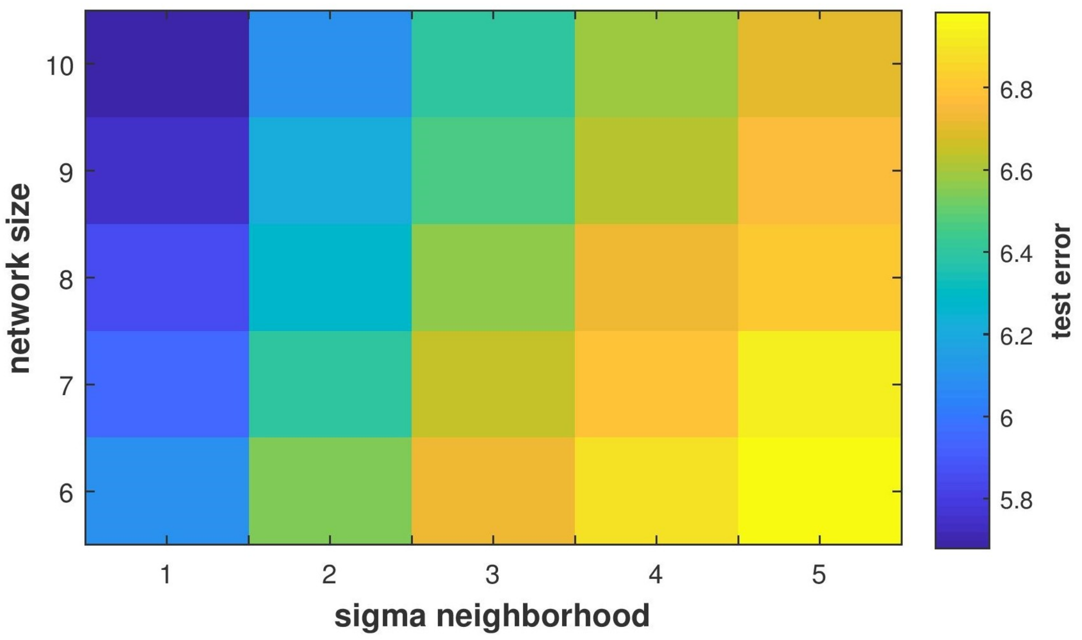


Figure 2.5: 5-folds cross-validation.