ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Unsupervised and reinforcement learning in neural networks
Week 6

Lecturer: Marc-Oliver Gewaltig

Assistants: Dane Corneil
Berat Denizdurduran

Corrections: Kohonen maps

Exercise 1: "Blob formation"

1.1. We consider the state evolution equation for the first neuron outside the blob. Since we are interested in the steady-state solution, we drop the time dependency and write $y(t+1) = y(t) = y$. Since the neuron we consider does not receive input from the x layer, it only receives d excitatory inputs from its close neighbours and $N - d$ inhibitory inputs from remote neighbours. Thus we have:

$$\begin{aligned} & y = g\left(d - \beta(N-d)\right) = 0 \\ \Rightarrow & d - \beta(N-d) < 0 \\ \Rightarrow & d < \beta N - \beta d \\ \Rightarrow & d(1+\beta) < \beta N \\ \Rightarrow & d(1+\frac{1}{\beta}) < N. \end{aligned}$$

Since $\beta \in (0, 1]$, $(1 + 1/\beta) \in [2, \infty)$, so that for any value of β, $N > 2d$ holds.

1.2. If $\beta = 1$, then $N > 2d$. Following the same argument as in 1.1, but for the last *active* neuron, we get $N \leq 2d + 2$ (still for $\beta = 1$). Thus $2d < N \leq 2d + 2$. If either of the inequalities is violated, the "limit" neurons of the blob will change their activity until both inequalities hold again.

1.3. For $\beta = 1$, $d = 2$ and $M = 10$, the evolution of active neurons looks like this:

y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}
0	0	1	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0

Exercise 2: Batch vs. online learning

No special trick here, just calculation, starting from the expression of the weight update at step $N + 1$:

$$
\begin{aligned}
\Delta\vec{w} &= \Delta\vec{w}^{N+1} - \Delta\vec{w}^{N} \\
&= \frac{1}{N+1}\sum_{\mu=1}^{N+1}(\vec{x}^{\mu} - \vec{w}_0) - \frac{1}{N}\sum_{\mu=1}^{N}(\vec{x}^{\mu} - \vec{w}_0) \\
&= \frac{1}{N+1}\left((\vec{x}^{N+1} - \vec{w}_0) + \sum_{\mu=1}^{N}(\vec{x}^{\mu} - \vec{w}_0)\right) - \frac{1}{N}\sum_{\mu=1}^{N}(\vec{x}^{\mu} - \vec{w}_0) \\
&= \frac{1}{N+1}(\vec{x}^{N+1} - \vec{w}_0) + \underbrace{\left(\frac{1}{N+1} - \frac{1}{N}\right)}_{\frac{-1}{N(N+1)}}\sum_{\mu=1}^{N}(\vec{x}^{\mu} - \vec{w}_0) \\
&= \frac{1}{N+1}\left((\vec{x}^{N+1} - \vec{w}_0) - \underbrace{\frac{1}{N}\sum_{\mu=1}^{N}(\vec{x}^{\mu} - \vec{w}_0)}_{\Delta\vec{w}^{N}}\right) \\
&= \eta(N)(\vec{x}^{N+1} - (\vec{w}^{0} + \Delta\vec{w}^{N}))\,,
\end{aligned}
$$

where $\eta(N) = 1/(N+1)$.

Exercise 3: Kohonen maps

3.1 For all machine learning algorithms, choosing the right parameter values is a crucial (and often somewhat empirical) step. Most parameters have a "sweet spot", where the algorithm performs as expected by the theoretical results, but below or above this range, the algorithm fails, or performs badly. In this exercise, we wanted you to experience first hand the performance of learning of a Kohonen map and the importance of choosing parameters. We give only a short explanation of the effects of each parameters: it is up to the reader to try this in practice.

The sigma controls the strength and range of the neighborhood effect of the points of the map. In the limit of small sigma, the algorithm is essentially the competitive learning algorithm, with possible dead units (very likely with our poor initialization). In the limit of big sigmas, all units learn the same thing, i. e. the center of the points.

3.2. Small learning rates cause the algorithm to converge slower, but the final solution will be more stable. However, one might get stuck in local minima. Large learning rates cause the algorithm to converge faster, "flying" over shallow local minima, but the weights might fail to converge at all.

3.3. A nice feature of the Kohonen map is that the neighborhood relationship helps to prevent overfitting. In many unsupervised algorithms, fitting the 100 points we are given with 100 weights would cause each weight to go to one point. However, we know that the true distribution of the points is uniform. In a sense, this knowledge is incorporated in the neighborhood relations of the Kohonen map. This causes it to fit the data with a more or less regular grid that captures the essence of the underlying distribution better than a naive clustering algorithm with so many free parameters would do.