Solutions for week 3
Hebbian rules and ICA

Exercise 1: Clustering

1.1 Show for competitive learning that every prototype is at the center of its group/cluster after convergence (i.e. in the steady state).

Use the (batch) learning rule: $\Delta\vec{w}_i = \eta \sum_{\mu\in C_i} (\vec{x}^\mu - \vec{w}_i)$.

Solution:

Convergence means $\Delta\vec{w}_i = \vec{0}$ for all i. So if we have convergence, we note $N_i = |C_i|$ the cardinal of group i, then we have

$$\eta \sum_{\mu\in C_i} (\vec{x}_\mu - \vec{w}_i) = \vec{0} \text{ so } \sum_{\mu\in C_i} \vec{x}_\mu - N_i\vec{w}_i = \vec{0} \text{ and finally } \vec{w}_i = \frac{1}{N_i}\sum_{\mu\in C_i} \vec{x}_\mu.$$

This is the definition of the "center of mass" of group i.

1.2 Consider now the online version of the learning rule: $\Delta\vec{w}_i = \eta(\vec{x}^\mu - \vec{w}_i)$. Calculate the fluctuation of the weight update in the steady state, i.e. its variance $\langle\Delta\vec{w}_i^2\rangle$.

Solution:

For the steady state, we know that $\langle\Delta\vec{w}_i\rangle = 0$. The variance is therefore given by:

$$\begin{aligned} Var(\Delta\vec{w}_i) &= \langle(\Delta\vec{w}_i)^2\rangle_\mu = \langle\eta^2(\vec{x}^\mu - \vec{w}_i))^2\rangle_\mu \\ &= \eta^2\langle(\vec{x}^\mu - \bar{x})^2\rangle_\mu = \eta^2 Var(\vec{x}). \end{aligned}$$

We see that the fluctuations are proportional to the variance of the associated data points.

Exercise 2: Batch vs. online learning

In the lecture, it was argued that batch learning and online learning are equivalent in the limit of small learning rates. Show that for competitive learning, batch and online learning are are also equivalent if the learning rate is decreased in the right way.

Let $\vec{w}_0$ be the initial weight before presentation of the input patterns $\vec{x}^\mu$. For simplicity, let us consider only patterns for which the considered weight is the winner. According to the batch rule, the weight change after the first N input patterns is then given by

$$\Delta\vec{w}^N = \frac{1}{N}\sum_{\mu=1}^{N}(\vec{x}^\mu - \vec{w}^0). \tag{1}$$

Calculate the difference between the weight vector after $N+1$ and after N input patterns and show that this leads to an online learning rule in which the weight change caused by pattern $N+1$ is given by:

$$\Delta\vec{w} = \eta(N)\left(\vec{x}^{N+1} - (\vec{w}^0 + \Delta\vec{w}^N)\right) = \eta(N)\left(\vec{x}^{N+1} - \vec{w}^N\right). \tag{2}$$

How does the learning rate η depend on N?

Solution:

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: Batch vs. online: case of clustering

Define the following loss function that minimizes the squared distance between each data point and its closest weight vector:

$$L(w_1, w_2...) = \frac{1}{2}\sum_{\mu} ||x^{\mu} - w_{j^*(\mu)}||^2$$

where $j^*(\mu)$ is the index of the closest weight vector for pattern μ.

3.1 Apply gradient descent (batch rule) with learning rate η.

Solution:

We denote $C_k = \{\mu :\ j^*(\mu) = k\}$, i.e. the set of all patterns with closest vector being $\vec{w}_k$. Then we do gradient descent of loss with respect to weight $\vec{w}_k$:

$$\begin{aligned}
\Delta\vec{w}_k &= -\eta\nabla_{\vec{w}_k} L \\
&= -\eta \sum_{\mu\in C_k}(\vec{w}_k - \vec{x}^{\mu}) \\
&= \eta \sum_{\mu\in C_k}(\vec{x}^{\mu} - \vec{w}_k)
\end{aligned}$$

3.2 Apply gradient descent (online rule) with learning rate η. What is the relation to competitive Hebbian learning?

Solution:

To make the learning rule online, we only do one update of one of the patterns μ:

$$\Delta\vec{w}_k = \eta\ (\vec{x}^{\mu} - \vec{w}_k)\cdot\delta_k^{j^*(\mu)}$$

where $\delta_k^{j^*(\mu)} = 1$ if $j^*(\mu) = k$ otherwise it is 0 (Dirac-delta function)
This is a hard competitive Hebbian learning because only the closest vector is updated. The other weight vectors are not updated due to strong lateral inhibition.

3.3 How can you choose a reduction scheme for η so that the result of online is exactly the result of batch? **Hint:** get inspiration from exercise 2.

Solution:

Using a similar idea as Exercise 2, we could compare weight updates of a batch of $N+1$ input and a batch of N input. Suppose the extra pattern in the $N+1$ batch is $\vec{x}^\nu$

$$\Delta\vec{w}_k^{N+1} - \Delta\vec{w}_k^N = \eta\ (\vec{x}^\nu - \vec{w}_k) \cdot \delta_k^{j^*(\nu)}$$

This is equivalent to the online rule of ν. So, there is no necessary reduction scheme for η. This is different from Exercise 2 because the update rule is not scaled by $\frac{1}{N}$ in this setup. If the loss function is scaled by $\frac{1}{N}$, then we could apply the result from exercise 2 to obtain $\eta(N) = 1/N$.

Exercise 4: Blob formation

Imagine a 1-dimensional recurrent network with M neurons and cyclic boundary conditions. The dynamics of the neurons is given by

$$y_i(t+1) = g\left(\sum_k w_{ik}x_k(t) + \sum_j B_{ij}y_j(t)\right)\ , \tag{3}$$

where B_{ij} are the recurrent weights. A neuron receives local excitation from itself and its d neighbours ($d \ll M$) on both sides: $B_{ij} = 1$ for $|i-j| \leq d$, and inhibition from all others: $B_{ij} = -\beta$ with $0 < \beta \leq 1$ otherwise. The activation function $g(h)$ is the Heaviside function, i. e. $g(h) = 1$ for $h \geq 0$ and $g(h) = 0$ for $h < 0$.

4.1 In class: Imagine that one single neuron is stimulated and therefore becomes active. This neuron will excite its neighbours and cause an "activity blob". Show that in the steady state of the network, the number N of active neurons is larger than $2d$. **Hint:** You may assume that the active neurons form a "blob" of neighbouring neurons. Consider the net input to the first neuron *outside* the blob and use the fact that this neuron is silent to find an inequality for N.

Solution:

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:

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

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

4.2 How does the strength of inhibition, i. e. the value of β, affect the number of active neurons? What can you say about how many neurons are active in the steady state in the limit $\beta = 1$? **Hint:** Consider the net input to the last neuron inside the blob to get another constraint on N.

Solution:

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.

4.3 Assume that the input to the third neuron is 1 (and there are no other external inputs to the network). Compute the first three time steps of the network dynamics for $\beta = 1$, $d = 2$ and $M = 10$. Assume that initially all neurons are silent.

Solution:

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 5:

5.1 In class: Show that if all prototypes/weight vectors $\vec{w}_k$ are normalized, choosing the nearest prototype $\vec{w}_i$ (i.e., the prototype for which $||\vec{x} - \vec{w}_i||^2 \leq ||\vec{x} - \vec{w}_k||^2 \; \forall k \neq i$) is equivalent to choosing the neuron i with the strongest input $\vec{w}_i^T \vec{x}$.

Solution:

We start by extending the squares on both sides of the inequality:

$$||\vec{x}||^2 - 2\vec{x}^T\vec{w}_i + ||\vec{w}_i||^2 \leq ||\vec{x}||^2 - 2\vec{x}^T\vec{w}_k + ||\vec{w}_k||^2.$$

Since the weight vectors are normalized, we have $||\vec{w}_k||^2 = 1, \forall k$, the corresponding terms on each side of the inequality. We can also eliminate the $||\vec{x}||^2$ that appear on each side. Dividing each side by -2 (without forgetting that dividing by a negative number causes the inequality to flip direction), we get

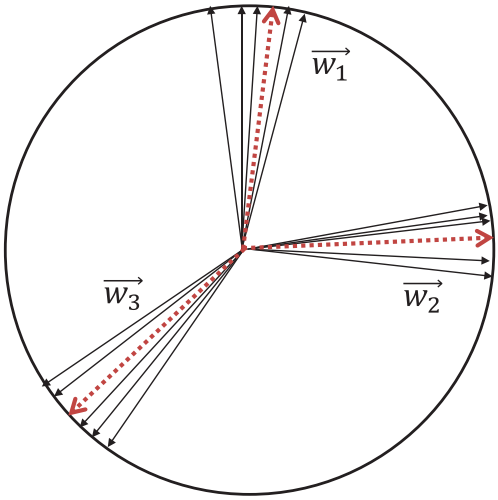
$$\vec{x}^T\vec{w}_i \geq \vec{x}^T\vec{w}_k.$$

This means that indeed, the nearest prototype is the one with the strongest input.

5.2 This question invites you to rethink what you saw in class: does the result still hold true if it is the data that is normalized (i. e. $||\vec{x}^\mu||^2 = 1 \, \forall \mu$), but the weight vectors are not? Consider the case when the clustering algorithm is close to convergence. (**Hint:** See image.)

Solution:

Looking at the derivation of 3.1. in isolation, the argument does not hold anymore when the weight vectors are not normalized, since the scalar product with a very large weight vector in an arbitrary direction could always be the largest, irrespective of its relation to the data samples. However, from exercise 1 we know, that if we are at the steady state (or close to it), the weight vector will be the center-of-mass of the cluster it represents and hence will be approximately



normalized when the data samples are:

$$| \vec{w} |^2 = | \frac{1}{N} \sum \vec{x}^\mu |^2 = \frac{1}{N^2} (\sum_{\mu,\nu} \vec{x}^{\mu,T} \vec{x}^\nu) = \frac{1}{N^2} (\sum_{\mu} \vec{x}^{\mu,T} \vec{x}^\mu + \sum_{\mu,\nu,\mu\neq\nu} \vec{x}^{\mu,T} \vec{x}^\nu)$$

$$= \frac{1}{N^2} (N + \sum_{\mu,\nu,\mu\neq\nu} \vec{x}^{\mu,T} \vec{x}^\nu) \approx \frac{1}{N^2} (N + \sum_{\mu,\nu,\mu\neq\nu} 1) = \frac{1}{N^2} (N + N(N-1)) = 1,$$

where we assumed that the scalar product between data points *belonging to the same cluster* are close to 1 and the data samples themselves are normalized.