LCPB 22-23 exercise 2 (Restricted Boltzmann Machines) & assignment

One group member will submit the Jupyter notebook, compiled with all figures, using the interface in moodle. Do not submit other files.

The notebook's name should be ***group23??_exerciseX.ipynb*** *where*

- *?? is the group's code*
- *X is the number of this exercise.*

The notebook should start with the list of students (Name, Family Name, matriculation number).

For the template tex of the assignment, see the drive.

The main aim of this exercise is to learn the correlations in the data by looking at the weights ($w_{i\mu}$) and biases learned by the RBM.

For the data to analyze, you can choose among:

(a) The data of CASE=1 (for one-hot encoding) introduced in the lesson.

(b) Data in DATA_c (new CASE=2), with one-hot encoding for A=6 and unknown structure. *Hint: in this case, M=2 hidden units with proper backward one-hot encoding (point 4) should do it.*

Check if one of the following points improve and/or stabilize the performance of the RBM.

1.
Use bits [0,1] or [-1,1]. Is reading weights easier in one of the two cases?

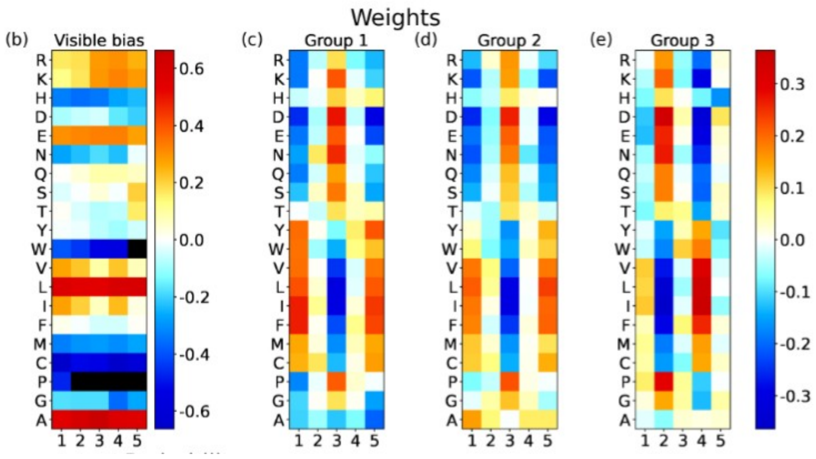
2.
Implement an Adam or RMSprop gradient descent algorithm in the RBM notebook.

3.
Increase the number of contrastive divergence steps.

4.
Implement a backward step of contrastive divergence that preserves the one-hot encoding structure, thus generating one of A states in a block rather than one of 2^A possible states. One randomly picks a block of A bits with probability proportional to the Boltzmann weight of its one-hot encoding, given the state of hidden units, the weights, and the local biases in that block.

OPTIONAL

5.
Plot the weights $w_{i\mu}$ of each hidden unit μ in an A×G table (as in the example on the right), with a color map preserving the rule that red means positive and blue means negative weight.



5.
Implement the "centering trick" described in the following pages, taken from Matteo Bortoletto's master thesis (note that different symbols for biases: a$\rightarrow$b, b$\rightarrow$c).

6.
Compute the log-likelihood $\boldsymbol{L}$ as a function of the epoch, or try one of the other quality indicators proposed in the paper by Decelle et al (drive), see especially the supplementary pdf. To compute the partition function (the hardest part in evaluating $\boldsymbol{L}$), in our case one needs $A^G*2^M \sim 10^5$ Boltzmann factors, which is doable.

4.1 The centering trick

The centering trick consists in shifting the visible and hidden variables by some offset parameters $\boldsymbol{\mu} = (\mu_1, \dots, \mu_N)$ and $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_M)$, respectively. The energy for the corresponding centered binary RBM is given by

$$E(\boldsymbol{v}, \boldsymbol{h}) = -(\boldsymbol{v} - \boldsymbol{\mu})^T \boldsymbol{b} - \boldsymbol{c}^T (\boldsymbol{h} - \boldsymbol{\lambda}) - (\boldsymbol{v} - \boldsymbol{\mu}) \mathbf{W} (\boldsymbol{h} - \boldsymbol{\lambda}). \tag{4.1}$$

For $\boldsymbol{\mu} = \boldsymbol{\lambda} = \mathbf{0}$ we recover the normal binary RBM, for $\boldsymbol{\mu} = \langle \boldsymbol{v} \rangle_d$ and $\boldsymbol{\lambda} = \mathbf{0}$ we obtain the original centering trick by Tang and Sutskever and for $\boldsymbol{\mu} = \langle \boldsymbol{v} \rangle_d$ and $\boldsymbol{\lambda} = \langle \boldsymbol{h} \rangle_d$ we obtain the model by Montavon and Müller.

The expressions for the factors of the conditional probabilities (3.10) are now given by

$$p(V_l = 1 | \boldsymbol{h}) = \sigma \left(\sum_{i=1}^{n} w_{il} (h_i - \lambda_i) + b_l \right), \tag{4.2}$$

$$p(H_l = 1 | \boldsymbol{v}) = \sigma \left(\sum_{j=1}^{n} w_{lj} (v_j - \mu_j) + c_l \right). \tag{4.3}$$

Again, we can find the RBM distribution over $\boldsymbol{V}$ by marginalizing (see Equation (2.12)):

$$\begin{aligned}
p(\boldsymbol{v}) &= \sum_{\boldsymbol{h}} p(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v}, \boldsymbol{h})} \\
&= \frac{1}{Z} \sum_{h_1} \sum_{h_2} \cdots \sum_{h_n} e^{\sum_{j=1}^{m} b_j (v_j - \mu_j)} \prod_{i=1}^{n} e^{(h_i - \lambda_i)\left(c_i + \sum_{j=1}^{m} w_{ij} (v_j - \mu_j)\right)} \\
&= \frac{1}{Z} e^{\sum_{j=1}^{m} b_j (v_j - \mu_j)} \prod_{i=1}^{n} \sum_{h_i} e^{(h_i - \lambda_i)\left(c_i + \sum_{j=1}^{m} w_{ij} (v_j - \mu_j)\right)} \\
&= \frac{1}{Z} \prod_{j=1}^{m} e^{b_j (v_j - \mu_j)} \prod_{i=1}^{n} \left(e^{-\lambda_i \left(c_i + \sum_{j=1}^{m} w_{ij} (v_j - \mu_j)\right)} + e^{(1 - \lambda_i)\left(c_i + \sum_{j=1}^{m} w_{ij} (v_j - \mu_j)\right)} \right) \\
&= \frac{1}{Z} \prod_{j=1}^{m} e^{b_j (v_j - \mu_j)} \prod_{i=1}^{n} e^{-\lambda_i \left(c_i + \sum_{j=1}^{m} w_{ij} (v_j - \mu_j)\right)} \left(1 + e^{c_i + \sum_{j=1}^{m} w_{ij} (v_j - \mu_j)} \right).
\end{aligned} \tag{4.4}$$

Finally, the log-likelihood gradients now take the form:

$$\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta} | \boldsymbol{v})}{\partial w_{ij}} \equiv \Delta w_{ij} = \langle (v_j - \mu_j)(h_i - \lambda_i) \rangle_d - \langle (v_j - \mu_j)(h_i - \lambda_i) \rangle_m \tag{4.5}$$

$$\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta} | \boldsymbol{v})}{\partial b_j} \equiv \Delta b_j = \langle v_j - \mu_j \rangle_d - \langle v_j - \mu_j \rangle_m = \langle v_j \rangle_d - \langle v_j \rangle_m \tag{4.6}$$

$$\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta} | \boldsymbol{v})}{\partial c_i} \equiv \Delta c_i = \langle h_i - \lambda_i \rangle_d - \langle h_i - \lambda_i \rangle_m = \langle h_i \rangle_d - \langle h_i \rangle_m, \tag{4.7}$$

which in vector form read

$$\Delta \mathbf{W} = \left\langle (\boldsymbol{v} - \boldsymbol{\mu})(\boldsymbol{h} - \boldsymbol{\lambda})^T \right\rangle_d - \left\langle (\boldsymbol{v} - \boldsymbol{\mu})(\boldsymbol{h} - \boldsymbol{\lambda})^T \right\rangle_m \tag{4.8}$$

$$\Delta \boldsymbol{b} = \langle \boldsymbol{v} \rangle_d - \langle \boldsymbol{v} \rangle_m \tag{4.9}$$

$$\Delta \boldsymbol{c} = \langle \boldsymbol{h} \rangle_d - \langle \boldsymbol{h} \rangle_m . \tag{4.10}$$

These equations show that centering only affects the gradient with respect to the weights. The important results is that it can be shown that the gradient of a centered RBM is invariant to flip transformations if a flip of v_j to $1 - v_j$ implies a change of μ_j to $1 - \mu_j$ and a flip of h_i to $1 - h_i$ implies a change of λ_i to $1 - \lambda_i$ [26]. This holds for $\mu_j = \lambda_i = 0.5$ but also for the expectation values of v_j and h_j under any distribution. Moreover, if the offsets are set to the expectation values, centered RBMs are also invariant to any shift of variables, not only to flip transformations [26].

Montavon and Müller have shown that centering improves the conditioning of the underlying optimization problem [27]. More precisely, the ratio between the highest and the lowest eigenvalue of the Hessian matrix is smaller. This ratio is known as *condition number* of the Hessian and it encodes how hard a strongly convex problem is to solve. Larger condition numbers imply slower convergence of gradient descent because in some directions the gradient will change rapidly and in others it will change very slowly.

4.2 Training centered Restricted Boltzmann Machines

If we set $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$ to the expected values of the variables, which is the most common choice, these values may depend on the model parameters and thus they may change during training. Therefore, we need to adapt the standard learning algorithm so that the offsets are updated to match the expectations under the new distribution we get after each parameter update. Moreover, when updating the offsets we need to transform the RBM parameters such that the probability distribution remains the same. An RBM with offsets $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$ can be transformed into an RBM with offsets $\tilde{\boldsymbol{\mu}}$ and $\tilde{\boldsymbol{\lambda}}$ by

$$\tilde{\mathbf{W}} = \mathbf{W}, \tag{4.11}$$

$$\tilde{\boldsymbol{b}} = \boldsymbol{b} + \mathbf{W}(\tilde{\boldsymbol{\lambda}} - \boldsymbol{\lambda}), \tag{4.12}$$

$$\tilde{\boldsymbol{c}} = \boldsymbol{c} + \mathbf{W}^T(\tilde{\boldsymbol{\mu}} - \boldsymbol{\mu}), \tag{4.13}$$

such that $E(\boldsymbol{v}, \boldsymbol{h}|\boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\lambda}) = E(\boldsymbol{v}, \boldsymbol{h}|\tilde{\boldsymbol{\theta}}, \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\lambda}}) + \mathit{const}$ is guaranteed. Clearly, these equations can be used to transform a centered RBM into a normal one and vice versa, emphasizing that normal and centered RBMs are just different parametrizations of the same model class.

Algorithm 3 shows the pseudo-code to train a centered RBM. Notice that $\langle \cdot \rangle$ denotes the average over the samples of the current batch. Thus, for example, $\langle \boldsymbol{v}_d \rangle$ is the average of the samples $\boldsymbol{v}_d$ in the current batch, which is taken as approximation of $\langle \boldsymbol{v} \rangle_d$. Similarly, $\langle \boldsymbol{h}_d \rangle = \langle p(\boldsymbol{h} = \mathbf{1}|\boldsymbol{v}_d) \rangle$

is an approximation for $\langle \boldsymbol{h} \rangle_d$. Note that the offsets are updated using a moving average with sliding factors $\zeta_\mu, \zeta_\lambda \in (0,1)$ (usually $\zeta \sim 0.01$). This is done to get a smoother approximation of the parameter updates in case the approximation of the mean values can be biased. For example, if we use the model mean or we have small mini-batch sizes the use of the moving average leads to stabler updates.

Algorithm 3 Training a centered RBM [26].

```
Input: RBM, data.
Output: Trained RBM.
 1: Initialize $\mathbf{W}$, $\boldsymbol{b}$, $\boldsymbol{c}$, $\boldsymbol{\mu}$, $\boldsymbol{\lambda}$, $\eta$ (learning rate), $\zeta_\mu$, $\zeta_\lambda$ (moving average factors).
 2: repeat
 3:   for all batch in data do
 4:     for all sample $\boldsymbol{v}$ in batch do
 5:       Compute $\boldsymbol{h}_d$, $\boldsymbol{v}_m$ and $\boldsymbol{h}_m$ using, for example, PCD or PT;
 6:     Estimate $\boldsymbol{\mu}_{batch} = \langle \boldsymbol{v}_d \rangle$ and $\boldsymbol{\lambda}_{batch} = \langle \boldsymbol{h}_d \rangle$
 7:     /* Transform the parameters with respect to the new offsets        */
 8:     $\boldsymbol{b} \leftarrow \boldsymbol{b} + \zeta_\lambda \mathbf{W}(\boldsymbol{\lambda}_{batch} - \boldsymbol{\lambda})$;
 9:     $\boldsymbol{c} \leftarrow \boldsymbol{c} + \zeta_\mu \mathbf{W}^T(\boldsymbol{\mu}_{batch} - \boldsymbol{\mu})$;
10:     /* Update the offsets using a moving average with factors $\zeta_\mu$ and $\zeta_\lambda$   */
11:     $\boldsymbol{\mu} \leftarrow (1-\zeta_\mu)\boldsymbol{\mu} + \zeta_\mu \boldsymbol{\mu}_{batch}$;
12:     $\boldsymbol{\lambda} \leftarrow (1-\zeta_\lambda)\boldsymbol{\lambda} + \zeta_\lambda \boldsymbol{\lambda}_{batch}$;
13:     /* Update the parameters according to the gradients                */
14:     $\Delta\mathbf{W} \leftarrow \left\langle (\boldsymbol{v}_d - \boldsymbol{\mu})(\boldsymbol{h}_d - \boldsymbol{\lambda})^T \right\rangle - \left\langle (\boldsymbol{v}_m - \boldsymbol{\mu})(\boldsymbol{h}_m - \boldsymbol{\lambda})^T \right\rangle$;
15:     $\Delta\boldsymbol{b} \leftarrow \langle \boldsymbol{v}_d \rangle - \langle \boldsymbol{v}_m \rangle$;
16:     $\Delta\boldsymbol{c} \leftarrow \langle \boldsymbol{h}_d \rangle - \langle \boldsymbol{h}_m \rangle$;
17:     $\mathbf{W} \leftarrow \mathbf{W} + \eta\Delta\mathbf{W}$;
18:     $\boldsymbol{b} \leftarrow \boldsymbol{b} + \eta\Delta\boldsymbol{b}$;
19:     $\boldsymbol{c} \leftarrow \boldsymbol{c} + \eta\Delta\boldsymbol{c}$;
20: until training is finished;
```

4.2.1 Centering the gradient

Instead of centering the parameters, we can also center the gradients, obtaining a "centered parameter update". This leads to the following updates [26]:

$$\Delta_c w_{ij} = \langle (v_j - \mu)(h_i - \lambda_i) \rangle_d - \langle (v_j - \mu_j)(h_i - \lambda_i) \rangle_m \tag{4.14}$$

$$\Delta_c b_j = \langle v_j \rangle_d - \langle v_j \rangle_m - \Delta_c w_{ij} \lambda_i \tag{4.15}$$

$$\Delta_c c_i = \langle h_i \rangle_d - \langle h_i \rangle_m - \Delta_c w_{ij}^T \lambda_j, \tag{4.16}$$

which in vector form read

$$\Delta\mathbf{W} = \left\langle (\boldsymbol{v} - \boldsymbol{\mu})(\boldsymbol{h} - \boldsymbol{\lambda})^T \right\rangle_d - \left\langle (\boldsymbol{v} - \boldsymbol{\mu})(\boldsymbol{h} - \boldsymbol{\lambda})^T \right\rangle_m \tag{4.17}$$

$$\Delta\boldsymbol{b} = \langle \boldsymbol{v} \rangle_d - \langle \boldsymbol{v} \rangle_m - \Delta_c \mathbf{W}\boldsymbol{\lambda} \tag{4.18}$$

$$\Delta\boldsymbol{c} = \langle \boldsymbol{h} \rangle_d - \langle \boldsymbol{h} \rangle_m - \Delta_c \mathbf{W}^T \boldsymbol{\mu}. \tag{4.19}$$

If we set $\boldsymbol{\mu} = (\langle \boldsymbol{v} \rangle_d + \langle \boldsymbol{v} \rangle_m)/2$ and $\boldsymbol{\lambda} = (\langle \boldsymbol{h} \rangle_d + \langle \boldsymbol{h} \rangle_m)/2$ we recover the enhanced gradient [26]. This confirms the fact that the enhanced gradient gradient is just a particular kind of centering trick. The pseudocode for training a normal RBM using the centered gradient is shown in Algorithm 4.

Algorithm 4 Training a normal RBM using the centered gradient [26].

```
Input: RBM, data.
Output: Trained RBM.
 1: Initialize $\mathbf{W}$, $\boldsymbol{b}$, $\boldsymbol{c}$, $\boldsymbol{\mu}$, $\boldsymbol{\lambda}$, $\eta$ (learning rate), $\zeta_\mu$, $\zeta_\lambda$ (moving average factors).
 2: repeat
 3:   for all batch in data do
 4:     for all sample $\boldsymbol{v}$ in batch do
 5:       Compute $\boldsymbol{h}_d$, $\boldsymbol{v}_m$ and $\boldsymbol{h}_m$ using, for example, PCD or PT;
 6:     Estimate $\boldsymbol{\mu}_{batch} = \langle \boldsymbol{v}_d \rangle$ and $\boldsymbol{\lambda}_{batch} = \langle \boldsymbol{h}_d \rangle$
 7:     /* Update the offsets using a moving average with factors $\zeta_\mu$ and $\zeta_\lambda$ */
 8:     $\boldsymbol{\mu} \leftarrow (1 - \zeta_\mu)\boldsymbol{\mu} + \zeta_\mu \boldsymbol{\mu}_{batch}$;
 9:     $\boldsymbol{\lambda} \leftarrow (1 - \zeta_\lambda)\boldsymbol{\lambda} + \zeta_\lambda \boldsymbol{\lambda}_{batch}$;
10:     /* Update the parameters using the centered gradient */
11:     $\Delta_c \mathbf{W} \leftarrow \left\langle (\boldsymbol{v}_d - \boldsymbol{\mu})(\boldsymbol{h}_d - \boldsymbol{\lambda})^T \right\rangle - \left\langle (\boldsymbol{v}_m - \boldsymbol{\mu})(\boldsymbol{h}_m - \boldsymbol{\lambda})^T \right\rangle$;
12:     $\Delta_c \boldsymbol{b} \leftarrow \langle \boldsymbol{v}_d \rangle - \langle \boldsymbol{v}_m \rangle - \Delta_c \mathbf{W} \boldsymbol{\lambda}$;
13:     $\Delta_c \boldsymbol{c} \leftarrow \langle \boldsymbol{h}_d \rangle - \langle \boldsymbol{h}_m \rangle - \Delta_c \mathbf{W}^T \boldsymbol{\mu}$;
14:     $\mathbf{W} \leftarrow \mathbf{W} + \eta \Delta_c \mathbf{W}$;
15:     $\boldsymbol{b} \leftarrow \boldsymbol{b} + \eta \Delta_c \boldsymbol{b}$;
16:     $\boldsymbol{c} \leftarrow \boldsymbol{c} + \eta \Delta_c \boldsymbol{c}$;
17: until training is finished;
```