1 Derivation RBM negative phase

The 'cost function' of the RBM is given by the Kullback-Leibler divergence,

$$C_{\boldsymbol{\lambda}} = D_{KL}(q||p_{\boldsymbol{\lambda}}) = \sum_{\boldsymbol{v}} q(\boldsymbol{v}) \log \left(\frac{q(\boldsymbol{v})}{p_{\boldsymbol{\lambda}}(\boldsymbol{v})} \right) = \sum_{\boldsymbol{v}} q(\boldsymbol{v}) \log(q(\boldsymbol{v})) - \sum_{\boldsymbol{v}} q(\boldsymbol{v}) \log(p_\lambda(\boldsymbol{v})) \tag{1}$$

The first term of the last equation is the Shannon entropy of the data distribution and is not dependend on the model paramters λ and will therefore not matter if we take the derivative with respect to these parameters. The second term of the last equation can be rewritten as,

$$-\sum_{\boldsymbol{v}} q(\boldsymbol{v}) \log(p_\lambda(\boldsymbol{v})) = -\langle \log(p_\lambda(\boldsymbol{v})) \rangle_{q(\boldsymbol{v})}, \tag{2}$$

which is the expectation value of the log-likelihood of $p_\lambda(\boldsymbol{v})$ w.r.t. the data distribution $q(\boldsymbol{v})$. The calculation can be simplified, by just maximizing the negative log-likelihood. Which means nothing else than making the training data the most likeli. The gradient of the likelihood (resp. KL-divergence) of a single training example $\boldsymbol{v}$ reads:

$$-\nabla_{\boldsymbol{\lambda}} \log(p_\lambda(\boldsymbol{v})) = \nabla_{\boldsymbol{\lambda}} \mathcal{E}_{\boldsymbol{\lambda}}(\boldsymbol{v}) + \nabla_{\boldsymbol{\lambda}} \log(Z_{\boldsymbol{\lambda}}) = \nabla_{\boldsymbol{\lambda}} \mathcal{E}_{\boldsymbol{\lambda}}(\boldsymbol{v}) + \frac{1}{Z_{\boldsymbol{\lambda}}} \nabla_{\boldsymbol{\lambda}} \sum_{\boldsymbol{v}',\boldsymbol{h}} e^{-E(\boldsymbol{v}',\boldsymbol{h})} = \tag{3}$$

$$\nabla_{\boldsymbol{\lambda}} \mathcal{E}_{\boldsymbol{\lambda}}(\boldsymbol{v}) - \frac{1}{Z_{\boldsymbol{\lambda}}} \sum_{\boldsymbol{v}',\boldsymbol{h}} e^{-E(\boldsymbol{v}',\boldsymbol{h})} \nabla_{\boldsymbol{\lambda}} E(\boldsymbol{v}',\boldsymbol{h}) = \nabla_{\boldsymbol{\lambda}} \mathcal{E}_{\boldsymbol{\lambda}}(\boldsymbol{v}) - \sum_{\boldsymbol{v}',\boldsymbol{h}} p_{\boldsymbol{\lambda}}(\boldsymbol{v}',\boldsymbol{h}) \nabla_{\boldsymbol{\lambda}} E(\boldsymbol{v}',\boldsymbol{h}) \tag{4}$$

Like in normal neural networks we would like to calculate the derivative for all data $\boldsymbol{v}$. But this is generally not feasible and we calculate it for batches $\mathcal{D}$ of the data distribution $q(\boldsymbol{v})$. It is very important to note, that the second term of the last equation is already summed over all v'. Therefore this part is already averaged over all possible configurations $\boldsymbol{v}$ and $\frac{1}{|\mathcal{D}|}\sum_{\boldsymbol{v}\in\mathcal{D}}$ cancels for this part.

$$-\nabla_{\boldsymbol{\lambda}} \langle \log(p_\lambda(\boldsymbol{v})) \rangle_{q(\boldsymbol{v})} \approx \frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{v}\in\mathcal{D}} \nabla_{\boldsymbol{\lambda}} \mathcal{E}_{\boldsymbol{\lambda}}(\boldsymbol{v}) - \sum_{\boldsymbol{v}',\boldsymbol{h}} p_\lambda(\boldsymbol{v}',\boldsymbol{h}) \nabla_{\boldsymbol{\lambda}} E(\boldsymbol{v}',\boldsymbol{h}) \tag{5}$$

In the second term we can again contract the variable $\boldsymbol{h}$ and we get:

$$-\nabla_{\boldsymbol{\lambda}} \langle \log(p_\lambda(\boldsymbol{v})) \rangle_{q(\boldsymbol{v})} \approx \frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{v}\in\mathcal{D}} \nabla_{\boldsymbol{\lambda}} \mathcal{E}_{\boldsymbol{\lambda}}(\boldsymbol{v}) - \sum_{\boldsymbol{v}} p_{\boldsymbol{\lambda}}(\boldsymbol{v}) \nabla_{\boldsymbol{\lambda}} \mathcal{E}(\boldsymbol{v}) = \tag{6}$$

$$\langle \nabla_{\boldsymbol{\lambda}} \mathcal{E}_{\boldsymbol{\lambda}}(\boldsymbol{v}) \rangle_{\mathcal{D}(\boldsymbol{v})} - \langle \nabla_{\boldsymbol{\lambda}} \mathcal{E}_{\boldsymbol{\lambda}}(\boldsymbol{v}) \rangle_{p_{\boldsymbol{\lambda}}(\boldsymbol{v})} \tag{7}$$