GROUPNAME – Title of project for the 2024-2025 assignment

James Brown, Barry White, and Simply Red
(Dated: March 30, 2025)

Log-likelihood To solve a problem using Bayesian method, as we do, we have to define

- the *likelihood function*, $p(X|\theta)$, that describes the probability of observing a dataset X given the value of parameters θ
- the *prior distribution*, $p(\theta)$, that describes the *a-priori* knowledge we have about the parameters.

These two are used to compute the *posterior distribution*

$$p(\theta|X) = \frac{p(X|\theta)\, p(\theta)}{\int d\theta' p(X|\theta')\, p(\theta')} \tag{1}$$

that describes the knowledge we have about parameters θ after observing the data X. As we will see the denominator of the posterior distribution in may cases is not possible to compute analytically. Markov Chain Monte Carlo methods are required to draw random samples of $p(\theta|X)$. The likelihood function is determined by the model and the measurement noise. Many generative models follows a *Maximum Likelihood Estimation* (MLE). In MLE parameters $\hat{\theta}$ that maximize the likelihood of generating observed data are chosen. Equivalently, the log-likelihood since log is monotonic.

$$\hat{\theta} = \arg_{\theta} \max \log p(X|\theta) \tag{2}$$

The most common approach used for training a generative model is to maximize the log-likelihood of the training dataset. By choosing the negative log-likelihood as the cost function, the learning procedure tries to find parameters that maximize the probability of the data. The log-likelihood $\ell_\theta(x)$ per data point x, averaged over M data points, gives the log-likelihood of data

$$\mathcal{L} = \frac{1}{M} \sum_{m \leq M} \ell_\theta\left(x^{(m)}\right) \tag{3}$$

[2] In training RBMs, our goal is to maximize the log-likelihood of the observed data x given the model parameters represented by a,b,w, respectively visible biases, hidden biases, weights.

$$\ell_\theta(x) = \ln \sum_z e^{-E(x,z)} - \ln \sum_{x'} \sum_z e^{-E(x',z)} \tag{4}$$

where the second therm is the partition function Z. The computation of the latter is intractable, the hard part resides in summing up the Boltmann weights of all possible configurations in Z, with D visible units and L hidden units, there are 2^{D+L} possible configurations. We followed instead the procedure suggested by Baiesi [1], that takes advantage of the energy function.

$$H_i(z) = a_i + \sum_\mu w_{i\mu} z_\mu \tag{5}$$

$$E(x,z) = -\sum_i H_i(z)\, x_i - \sum_\mu b_\mu z_\mu \tag{6}$$

$$e^{-E(x,z)} = \prod_\mu e^{b_\mu z_\mu} \prod_i e^{H_i(z) x_i} \tag{7}$$

in eq. 7 the first factor is the hidden units contribution to the energy, defined as $G(z)$. With this we can reach a reduced partition function $Z(z)$ defined as

$$Z(z) = G(z) \prod_i \left(1 + e^{H_i(z)}\right) \tag{8}$$

This is easy to compute and becomes numerically stable limiting the argument to avoid overflow. Since we used low value of L in our RBM we can compute the partition function at the start of the training

$$\ln Z = \ln \left[\sum_z G(z) \prod_{i=1}^{D} \left(1 + e^{H_i(z)}\right) \right] \tag{9}$$

then we averaged it over $x^{(m)}$ points of the dataset to get $\mathcal{L}$. This computation of the log-likelihood is used to identify the best models after a random search. The $\mathcal{L}$ behaves well with Bernoulli variables $\{0, 1\}$ leading to values that reach ~ -140 for our best models.

[1] Marco Baiesi **Log-likelihood computation for restricted Boltzmann machines**, 1–2 (2025).

[2] P. Mehta, M. Bukov, et al. **A high-bias, low-variance introduction to Machine Learning for physicists**, 21 (2018).