Notes on unsupervised learning with matrix product states

Erdong Guo and Pan Zhang
Institute of Theoretical Physics, Chinese Academy of Sciences

Abstract

In this note, an unsupervised learning algorithm by matrix product states (MPS) is proposed. We construct the energy function (cost function) of the MPS machine, and gradient descent method is used to optimize it. The analytical solution of the stable point equation is found in the $D = 1$ case, and factorized solutions in any bond dimension are studied. The numerical experiments show that the MPS machine actually works well.

1 Introduction

In the unsupervised learning, the task is to train a generative model which is able to output given data (images, configurations, etc.) with probability as large as possible. Many generative models have been proposed, include classic ones such as the Hopfield model and more successful ones e.g. the restricted Boltzmann machine and recently proposed generative adversary networks.

Here we are interested in possibility of powerful generative models using quantum systems. A natural choice is the wave function, which has a simple meaning of probability amplitude. If we further restrict the amplitude to be real, for example by letting parameters of the wave function to be real, then it simply denotes the squared root of probability.

Using language of tensor networks, we represent a wave function as a big tensor with real elements. Our aim is to train parameters of the tensor (in a variation form) such that the probability amplitude is maximized at configurations represented by given data.

The simplest tensor network is the matrix product state, where the big tensor is assumed to be composed as product of many three-way tensors.

2 Set-up

We consider a tensor network with n spins with configuration space

$$\chi = \{0,1\}^n = \underbrace{\{0,1\} \times \{0,1\} \cdots \times \{0,1\}}_{n}.$$

The given data such as black-white images in MNIST can be treated as spin configurations:

$$S = \{s_i^{(t)} \mid s_i^{(t)} \in \{0,1\}, t \in \{1,2,3\cdots T\}, i \in \{1,2,3,\cdots,n\}\}$$

where T denotes number of configurations (e.g. images) in the given data.

The likelihood, i.e. the generating probability, of given data by the MPS model is written as

$$P(S) = \frac{1}{Z^T}\prod_{t=1}^{T}\left[\sum_{\alpha_1=1}^{D}\sum_{\alpha_2=1}^{D},\dots,\sum_{\alpha_{n-1}=1}^{D} A^{(1)}_{s_1\alpha_1}A^{(2)}_{s_2\alpha_1\alpha_2}\cdots A^{(i)}_{s_i\alpha_{i-1}\alpha_i}\cdots A^{(n)}_{s_n\alpha_{n-1}}\Phi^{s_1^{(t)}s_2^{(t)}\cdots s_n^{(t)}}(s^{(t)})\right]^2,$$

where D is the bond dimension and

$$Z = \sum_{x\in\chi}\left[\sum_{\alpha_1,\alpha_2,\dots,\alpha_{n-1}} A^{(1)}_{x_1\alpha_1}A^{(2)}_{x_2\alpha_1\alpha_2}\cdots A^{(j+1)}_{x_{j+1}\alpha_j\alpha_{j+1}}\cdots A^{(n)}_{x_n\alpha_{n-1}}\Phi^{x_1x_2\cdots x_n}(x)\right]^2,$$

is the normalization factor (partition function), each component $A^{(i)}$ is a matrix if $i = 1$ or $i = n$, and a 3−way tensor otherwise.

In our case we assume our data is binary, which gives

$$\Phi^{s_1 s_2 \cdot s_n}(x) := \phi^{s_1}(x_1) \otimes \phi^{s_2}(x_2) \cdots \otimes \phi^{s_n}(x_n), x := (x_1, x_2, x_3, \cdots, x_n) \in \chi.$$

$$\phi^{s_i} : \{0,1\} \to \{(0,1),(1,0)\}, i \in \{1, 2 \cdots n\}, s_i \in \{0,1\},$$

Then the likelihood and partition function can be rewritten as product of vectors and matrices

$$P(S) = \frac{1}{Z^T} \prod_{t=1}^{T} \left[\prod_i A^{(i)}_{s_i^{(t)}} \right]^2$$

$$Z = \sum_{x \in \chi} \left[\prod_i A^{(i)}_{x_i} \right]^2 ./ \tag{1}$$

Here $A^{(i)}_{s_i}$ is either a vector (with $i = 1$ and $i = n$) or a matrix (with $i \neq i, n$).

The log-likelihood is written as

$$\mathcal{L}(S) = \log P(S) = 2 \sum_{t=1}^{T} \log \prod_{i=1}^{n} A^{(i)}_{s_i^{(t)}} - T \log Z. \tag{2}$$

Keep in mind that the product in the right hand side of the last equation is product of vectors and matrices, hence the log() function does not go into the product.

The most straight forward approach of log-likelihood maximization is gradient descent. The gradient can be computed as

$$\frac{1}{2} \frac{\partial \mathcal{L}}{\partial A^{(k)}_{\mu ij}} = \sum_{t=1}^{T} \delta(s_k^{(t)}, \mu) \frac{B_i^{(s,k)} C_j^{(s,k)}}{\sum_{i=1}^{D} \sum_{j=1}^{D} B_i^{(s,k)} A^{(k)}_{\mu ij} C_j^{(s,k)}} - T \sum_{x \in \chi} P(x) \delta(x_k, \mu) \frac{B_i^{(x,k)} C_j^{(x,k)}}{\sum_{i=1}^{D} \sum_{j=1}^{D} B_i^{(x,k)} A^{(k)}_{\mu ij} C_j^{(x,k)}}. \tag{3}$$

Here we use the shorthand notations as follows,

$$\begin{aligned} B_i^{(s,k)} &= A^{(1)}_{s_1} A^{(2)}_{s_2} \cdots A^{(k-1)}_{s_k \alpha_{k-1} i}, \\ C_j^{(s,k)} &= A^{(k+1)}_{s_{k+1} j \alpha_{k+1}} A^{(k+2)}_{s_{k+2} \alpha_{k+1} \alpha_{k+2}} \cdots A^{(n)}_{s_n}. \end{aligned}$$

For the initial conditions, a reasonable one is a small constant or random value.

We have to be very careful about the instability of learning, as when the mean value of the big tensor grows beyond 1, it will probably grow forever very quickly, making learning diverge.

3 The special case of $D = 1$

With bond dimension equals to 1, each 3-way tensor $A^{(i)}$ in MPS reduces to a vector of 2 components, denoted by $A^{(i)}_1$ and $A^{(i)}_0$.

Then the log-likelihood can be simplified to

$$\begin{aligned} \mathcal{L}(S) &= 2 \sum_{t=1}^{T} \sum_{i}^{n} \log A^{(i)}_{s_i^{(t)}} - T \log Z, \\ &= 2 \sum_{t=1}^{T} \sum_{i}^{n} \log A^{(i)}_{s_i^{(t)}} - T \sum_i \log[(A^{(i)}_0)^2 + (A^{(i)}_1)^2], \end{aligned} \tag{4}$$

The gradient is written as

$$\begin{aligned} \frac{1}{2} \frac{\partial \mathcal{L}}{\partial A^{(k)}_{\mu}} &= \sum_{t=1}^{T} \delta(s_k^{(t)}, \mu) \frac{1}{A^{(k)}_{\mu}} - T \sum_i \delta(i,k) \frac{A^{(i)}_{\mu}}{(A^{(i)}_0)^2 + (A^{(i)}_1)^2}, \\ &= \frac{T \rho^{(k)}_{\mu}}{A^{(k)}_{\mu}} - T \frac{A^{(k)}_{\mu}}{(A^{(k)}_0)^2 + (A^{(k)}_1)^2}, \end{aligned} \tag{5}$$

where $\rho_\mu^{(k)} = \frac{1}{T}\sum_{t=1}^{T}\delta(s_k^{(t)},\mu)$.

At the stationary point of gradient descent, we have

$$\frac{T\rho_\mu^{(k)}}{A_\mu^{(k)}} = \frac{TA_\mu^{(k)}}{(A_0^{(k)})^2 + (A_1^{(k)})^2},$$

which evaluates to

$$A_\mu^{(k)} = \sqrt{\rho_\mu^{(k)}}.$$

The last equation has a simple meaning that each element simply represents the density of black and white of each pixel in the given images.

4 Factorized solution

With $D > 1$ it seems that there should be always a simple solution which mimics $D = 1$ case in such a way that $A_{\mu ij}^{(k)}$ are identical given k and μ. In other words, components of each of two matrices of a three-way tensor are identical, and are equal to $A_\mu^{(k)}$ in $D = 1$ case where the value denotes the density.

Here we find the three way tensor $A_{\mu ij}^{(k)}$ written as $A_\mu^{(k)}\delta_{ij}$ might also be the solution. The log-likelihood can be written as follows:

$$\begin{aligned}\mathcal{L}(S) &= \mathcal{L}_1 - \mathcal{L}_2 \\ &= 2\sum_{s\in S}\log(A_{s_1\alpha_1}^{(1)}A_{s_2\alpha_1\alpha_2}^{(2)}\cdots A_{s_n\alpha_{n-1}}^{(n)}) - T\log(Z)\end{aligned} \tag{6}$$

At first, let us take the anzat of the solution into the partition and make some simplification as follows:

$$\begin{aligned}Z &= \sum_{x\in\mathcal{X}}(A_{s_1\alpha_1}^{(1)}\cdots A_{s_n\alpha_{n-1}}^{(n)}\Phi^{s_1s_2\cdots s_n})^2 \\ &= \sum_{x\in\mathcal{X}}(A_{s_1\alpha_1}^{(1)}A_{s_n\alpha_{n-1}}^{(n)}\delta_{\alpha_1\alpha_{n-1}}A_{s_2}^{(2)}\cdots A_{s_{n-1}}^{(n-1)}\Phi^{s_1s_2\cdots s_n})^2 \\ &= \sum_{s_1,s_n}(A_{s_1\alpha_1}^{(1)}A_{s_n\alpha_{n-1}}^{(n)}\delta_{\alpha_1\alpha_{n-1}}\Phi^{s_1s_{n-1}})\prod_{i=2}^{n-1}[(A_0^{(i)})^2 + (A_1^{(i)})^2]\end{aligned} \tag{7}$$

Then, the log likelihood could be written as:

$$\log(Z) = \log(\sum_{s_1,s_n}(A_{s_1\alpha_1}^{(1)}A_{s_n\alpha_{n-1}}^{(n)}\delta_{\alpha_1\alpha_{n-1}}\Phi^{s_1s_{n-1}})) + \sum_{i=2}^{n-1}\log[(A_0^{(i)})^2 + (A_1^{(i)})^2] \tag{8}$$

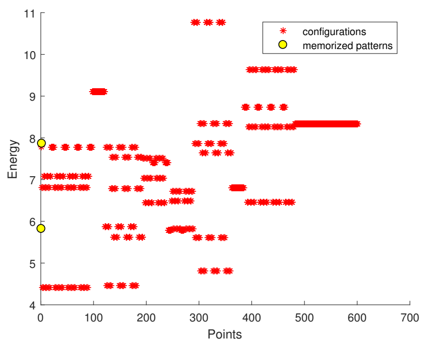
The gradient of the data term and system term could be calculated,

$$\begin{aligned}\frac{1}{2}\frac{\partial\mathcal{L}_1}{\partial A_{s_l\alpha_{k-1}\alpha_k}^{(k)}} &= \sum_{s\in S}\frac{A_{s_1\alpha_1}A_{s_2}\delta_{\alpha_1\alpha_2}\cdots A_{s_{k-1}}\delta_{\alpha_{k-2}\alpha_{k-1}}\cdots A_{s_{k+1}}\delta_{\alpha_k\alpha_{k+1}}\cdots A_{s_n\alpha_{n-1}}\delta_{s_ls_k}}{A_{s_1\alpha_1}A_{s_2}\delta_{\alpha_1\alpha_2}\cdots A_{s_{n-1}}\delta_{\alpha_{n-2}\alpha_{n-1}}A_{s_n\alpha_{n-1}}} \\ &= \sum_{s\in S}\frac{A_{s_1\alpha_{k-1}}A_{s_n\alpha_k}\Phi^{s_1s_n}}{A_{s_1\alpha_1}A_{s_n\alpha_{n-1}}\delta_{\alpha_1\alpha_{n-1}}\Phi^{s_1s_n}A_{s_k}}\delta_{s_ks_l}\end{aligned} \tag{9}$$

$$\begin{aligned}\frac{1}{2}\frac{\partial\mathcal{L}}{\partial A_{s_l\alpha_{k-1}\alpha_k}^{(k)}} &= \sum_{s\in S}\frac{A_{s_1\alpha_{k-1}}A_{s_n\alpha_k}}{A_{s_1\alpha_1}A_{s_n\alpha_{n-1}}\delta_{\alpha_1\alpha_{n-1}}A_{s_k}}\delta_{s_ks_l} - T\frac{A_{s_l}^k}{(A_0^{(k)})^2 + (A_1^{(k)})^2} \\ &= \frac{1}{A_{s_l}^{(k)}}\sum_{s\in S}\frac{A_{s_1\alpha_{k-1}}A_{s_n\alpha_k}}{A_{s_1\alpha_1}A_{s_n\alpha_{n-1}}\delta_{\alpha_1\alpha_{n-1}}}\delta_{s_ks_l} - T\frac{A_{s_l}^k}{(A_0^{(k)})^2 + (A_1^{(k)})^2}\end{aligned} \tag{10}$$

So we could construct a series of solution by two kinds of 3-way tensors: the diagonal matrix and the matrix whose elements are all identical.

(We can also assume the matrix has some certain symmetries. For instance, we can consider the case that A_{ij}^μ is diagonal but it has different diagonal elements.)



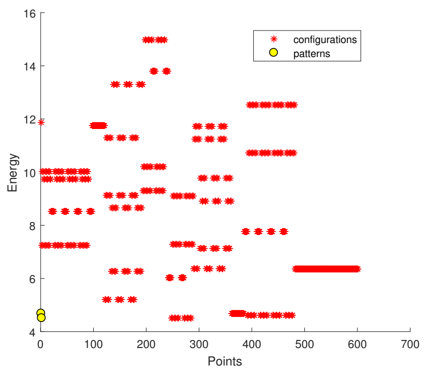


Figure 1: Right figure shows the energy of five dimensional vectors whose components are 1 or 0. Yellow points represent the data which would be memorized by the MPS, and the red stars represent other possible configurations of the input data. Left figure shows the energy of all vectors after training, which indicates that the certain vectors are actually memorized.

5 Memorizing two or three patterns

To get a sense of how this learning algorithm works, we analyzed the result when $D = 5, n = 5$.