Homework 3

Due date: Thursday, April 18, 2019

*Submit **a single .zip file** online using the link in the PSI wiki. You must submit all of your code as well as a .pdf summary that contains and discusses all relevant plots. Acknowledge any references you use as well as any other students with whom you collaborate.*

1 Identifying phase transitions using t-SNE

The objective of this problem is to use t-distributed stochastic neighbour embedding (t-SNE) to identify phase transitions based on spin configuration data.

As discussed in Lecture 8, t-SNE is a method that takes a dataset of high-dimensional vectors $\{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_m\}$ (with each $\mathbf{x}_i \in \mathbb{R}^N$ and N large) and creates a lower-dimensional representation of the dataset $\{\mathbf{x}'_1, \mathbf{x}'_2, \ldots \mathbf{x}'_m\}$. Here each $\mathbf{x}'_i \in \mathbb{R}^2$ or $\mathbb{R}^3$ such that the representation can be visualized.

Here, you will apply t-SNE to the problem of identifying phases of the two-dimensional classical Ising model and the classical Ising gauge theory. Your datasets will consists of spin configurations from these models at various temperatures. You may start from Laurens van der Maaten's Python implementation of t-SNE for the MNIST dataset of handwritten digits (see Reference [1]). You are also encouraged to look at the "Examples" section of this reference to see the results of applying t-SNE to other datasets.

Submit your code as well as a written summary (including relevant plots) of your answers to the following points.

a) Give a brief explanation of why the problem of using t-SNE to identify phases of matter from spin configurations is classified as unsupervised learning as opposed to supervised learning.

b) Look through the code from Reference [1]. Within it, there is a function `tsne` that takes arguments `X`, `no_dims`, `initial_dims` and `perplexity`. Explain the meaning of each of these arguments.

   Hint: For the perplexity, you may find it useful to look at Reference [2].

c) Modify the code from Reference [1] such that it applies t-SNE to the datasets from Tutorial 4 for the two-dimensional classical Ising model and the classical Ising gauge theory. Within your summary, include your results for each model at $L = 20$, 40 and 80.

   Hint: When applying the code from Reference [1] to the datasets from Tutorial 4, you may run into numerical issues due to the variable `sumP` within the function `Hbeta` sometimes becoming very small. If you run into these issues, try replacing the line `sumP = sum(P)` with `sumP = max(sum(P),1e-13)`.

d) Explain your results from part (c), including discussion of the following points:

- How do the clustering results from t-SNE change as L increases?
- Show that if you apply the t-SNE code twice to the same dataset, you get different results. Why do such differences occur?
- Compare your results for the Ising model with Figure 6 of Reference [3]. In this figure, t-SNE was applied to the two-dimensional Ising model on a lattice with $L = 30$. Do you observe similar behaviour?
- Compare your results with the results of Tutorial 4, which studied the same problem of phase identification using the more limited (linear) method known as principal component analysis. Discuss any differences that you observe.

2 Causality in reinforcement learning

Recall from Lecture 9 that in reinforcement learning, we aim to find a policy π that maximizes the expected cumulative reward J, which is given by

$$J(\pi) = \langle r(\tau) \rangle_{\tau \sim p_\pi(\tau)} ,$$

where $r(\tau)$ is the cumulative reward for the (Markovian) trajectory τ and $p_\pi(\tau)$ is the probability associated with this trajectory. A trajectory is defined as a series of states s_i and actions a_i such that

$$\tau \equiv \tau_1^T = s_1 \to a_1 \to s_2 \to a_2 \cdots \to a_{T-1} \to s_T,$$

where T is the terminal time. For a more general segment of this trajectory from starting time t_1 to ending time $t_2 > t_1$,

$$\tau_{t_1}^{t_2} = s_{t_1} \to a_{t_1} \to s_{t_1+1} \to a_{t_1+1} \cdots \to a_{t_2-1} \to s_{t_2} \to a_{t_2},$$

where we have assumed that all actions a_T on the terminal state s_T return us to s_T so that the above two definitions are consistent. We can decompose the cumulative reward along any such trajectory $\tau_{t_1}^{t_2}$ as

$$r(\tau_{t_1}^{t_2}) = \sum_{t'=t_1}^{t_2} r(a_{t'}, s_{t'}).$$

Let us control our choice of policy through some set of parameters θ such that $\pi \equiv \pi_\theta$. We can adjust these parameters using a learning algorithm such as gradient ascent, whereby $\theta \leftarrow \theta + \eta \nabla_\theta J(\pi_\theta)$ and η is the so-called *learning rate*.

Recall that the probability $p_{\pi_\theta}(\tau)$ associated with trajectory τ for policy π_θ can be expressed as

$$p_{\pi_\theta}(\tau) = p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t \mid s_t)\, p(s_{t+1} \mid s_t, a_t),$$

where $\pi_\theta(a_t \mid s_t)$ is the probability that our policy will choose action a_t given state s_t. The probability $p(s_{t+1} \mid s_t, a_t)$ is defined by the environment and is independent of the parameters θ. For more general trajectories $\tau_{t_1}^{t_2}$, we have

$$p_{\pi_\theta}(\tau_{t_1}^{t_2} \mid s_{t_1}) = \prod_{t'=t_1}^{t_2} \pi_\theta(a_{t'} \mid s_{t'})\, p(s_{t'+1} \mid s_{t'}, a_{t'})$$

and

$$p_{\pi_\theta}(\tau_{t+1}^{t_2} \mid s_t, a_t) = p(s_{t+1} \mid s_t, a_t)\; p_{\pi_\theta}(\tau_{t+1}^{t_2} \mid s_{t+1}).$$

Note that, by making use of the Markovian nature of our trajectory, one can write

$$p(s_t \mid \tau_{t_1}^{t-1}) = p(s_t \mid s_{t-1}, a_{t-1}).$$

In this question, we will examine the form of the gradients $\nabla_\theta J(\pi_\theta)$ using the causal state-action structure of our Markovian trajectories.

a) Show that the gradients $\nabla_\theta J(\pi_\theta)$ can be expressed as a sum of two terms such that

$$\nabla_\theta J(\pi_\theta) = \left\langle \sum_{t=2}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\; r(\tau_1^{t-1}) \right\rangle_{\tau \sim p_{\pi_\theta}(\tau)} + \left\langle \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\; r(\tau_t^{T}) \right\rangle_{\tau \sim p_{\pi_\theta}(\tau)}.$$

b) By using the fact that the probability $p_{\pi_\theta}(\tau)$ can be decomposed as

$$p_{\pi_\theta}(\tau) = p(s_1)\; p_{\pi_\theta}(\tau_1^{t-1}|s_1)\; p(s_t \mid \tau_1^{t-1})\; \pi_\theta(a_t \mid s_t)\; p_{\pi_\theta}(\tau_{t+1}^{T} \mid s_t, a_t),$$

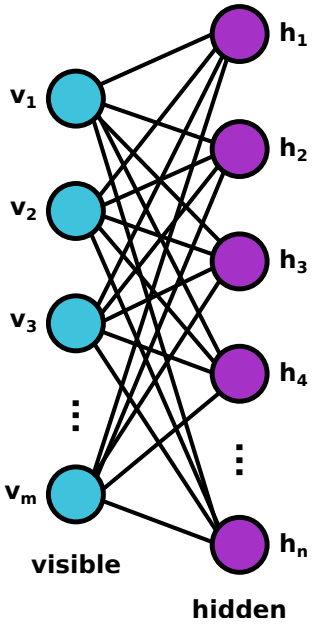
show that the first term in the expression from part a) is zero such that

$$\nabla_\theta J(\pi_\theta) = \left\langle \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\; r(\tau_t^{T}) \right\rangle_{\tau \sim p_{\pi_\theta}(\tau)}.$$

As a result of this simplification, we can effectively remove "noise" from our calculation of the gradients by limiting our consideration to future rewards.

3 Restricted Boltzmann machines

As seen in Lectures 11–13 and Tutorial 5, the architecture of a restricted Boltzmann machine (RBM) can be represented as



where $\mathbf{v} = (v_1, v_2, \ldots v_m)^T$ consists of m visible units with $v_i \in \{0, 1\}$ and $\mathbf{h} = (h_1, h_2, \ldots h_n)^T$ consists of n hidden units with $h_j \in \{0, 1\}$. The weights of the RBM are denoted as W_{ij}. The biases are denoted as b_i for the visible units and c_j for the hidden units. In this problem we use λ to denote all model parameters such that $\lambda = \{W, b, c\}$.

The probability distribution associated with the RBM is given by

$$p_\lambda(\mathbf{v}, \mathbf{h}) = \frac{1}{Z_\lambda} e^{-E_\lambda(\mathbf{v}, \mathbf{h})},$$

where

$$E_\lambda(\mathbf{v}, \mathbf{h}) = -\sum_{i=1}^{m} b_i \, v_i - \sum_{j=1}^{n} c_j \, h_j - \sum_{ij} W_{ij} \, v_i \, h_j,$$

and

$$Z_\lambda = \sum_{\mathbf{v}, \mathbf{h}} e^{-E_\lambda(\mathbf{v}, \mathbf{h})}.$$

Recall from Lecture 12 that each hidden variable h_i is independent of the other hidden variables $\mathbf{h}_{-i} \equiv \{h_1, \ldots, h_{i-1}, h_{i+1}, \ldots, h_n\}$ such that the conditional probabilities satisfy

$$p_\lambda\left(\mathbf{h} \mid \mathbf{v}\right) = \prod_{j=1}^{n} p_\lambda\left(h_j \mid \mathbf{v}\right).$$

Similarly, for the visible units

$$p_\lambda\left(\mathbf{v} \mid \mathbf{h}\right) = \prod_{i=1}^{m} p_\lambda\left(v_i \mid \mathbf{h}\right).$$

In this problem, we will derive explicit expressions for these conditional distributions in terms of the model parameters λ. We will then discuss the training and sampling procedures for generative modelling with RBMs.

a) Trace over the hidden variables $\mathbf{h}$ in order to show that

$$p_\lambda(\mathbf{v}) = \sum_{\mathbf{h}} p_\lambda(\mathbf{v}, \mathbf{h}) = \frac{1}{Z_\lambda} \prod_{i=1}^{m} e^{b_i v_i} \prod_{j=1}^{n} \left(1 + e^{c_j + \sum_i v_i W_{ij}}\right).$$

Similarly, show that the distribution over the hidden variables satisfies

$$p_\lambda(\mathbf{h}) = \frac{1}{Z_\lambda} \prod_{j=1}^{n} e^{c_j h_j} \prod_{i=1}^{m} \left(1 + e^{b_i + \sum_j W_{ij} h_j}\right).$$

b) Use your results from part a) as well as the normalization condition

$$p_\lambda(h_j = 1 \mid \mathbf{v}) + p_\lambda(h_j = 0 \mid \mathbf{v}) = 1$$

in order to show that

$$p_\lambda(h_j = 1 \mid \mathbf{v}) = \sigma\left(\sum_{i=1}^{m} v_i W_{ij} + c_j\right).$$

Similarly, show that

$$p_\lambda(v_i = 1 \mid \mathbf{h}) = \sigma\left(\sum_{j=1}^{n} W_{ij} h_j + b_i\right),$$

where, in the two above expressions, $\sigma(z) = 1/(1 + e^{-z})$ is the so-called *sigmoid* function.

c) Using 500 words or fewer, summarize the concepts discussed in lecture relevant to training an RBM. Your summary should be qualitative in nature and should contain at most two equations.

d) Let us now consider sampling from an RBM in order to calculate expectation values $\langle Q \rangle_{p_\lambda(\mathbf{v},\mathbf{h})}$ for some quantity Q. Within the block Gibbs Monte Carlo sampling procedure, we start from a visible sample $\mathbf{v}_0$, generate a new hidden sample $\mathbf{h}_0$ using the conditional distribution $p_\lambda(\mathbf{h} \mid \mathbf{v}_0)$, then generate a new visible sample $\mathbf{v}_1$ using the conditional distribution $p_\lambda(\mathbf{v} \mid \mathbf{h}_0)$, and then iterate this procedure for k step in order to produce the chain

$$\mathbf{v}_0 \to \mathbf{h}_0 \to \mathbf{v}_1 \to \mathbf{h}_1 \to \cdots \to \mathbf{v}_k \to \mathbf{h}_k.$$

Recall from Lecture 5 on Monte Carlo simulation that the transition probabilities $T_\lambda(\mu \to \nu)$ corresponding to transitions between states μ and ν (with $\mu = (\mathbf{v}, \mathbf{h})$ here) within the model parameterized by λ must satisfy detailed balance such that

$$\frac{T_\lambda(\mu \to \nu)}{T_\lambda(\nu \to \mu)} = \frac{p_\lambda(\nu)}{p_\lambda(\mu)}.$$

If we split the transition probability $T_\lambda(\mu \to \nu)$ into a product of a selection probability $g_\lambda(\mu \to \nu)$ and an acceptance probability $A_\lambda(\mu \to \nu)$, and then make use of the Metropolis algorithm, then we saw that the acceptance probabilities are given by

$$A_\lambda(\mu \to \nu) = \min\left(1, \frac{p_\lambda(\nu)}{p_\lambda(\mu)} \frac{g_\lambda(\nu \to \mu)}{g_\lambda(\mu \to \nu)}\right).$$

In the chain described above, we consider moves $(\mathbf{v}, \mathbf{h}) \to (\mathbf{v}', \mathbf{h})$ with selection probability $p_\lambda(\mathbf{v}' \mid \mathbf{h})$, as well as moves $(\mathbf{v}, \mathbf{h}) \to (\mathbf{v}, \mathbf{h}')$ with selection probability $p_\lambda(\mathbf{v}' \mid \mathbf{h})$. Show that proposed moves are always accepted within this procedure such that

$$A_\lambda((\mathbf{v}, \mathbf{h}) \to (\mathbf{v}', \mathbf{h})) = A_\lambda((\mathbf{v}, \mathbf{h}) \to (\mathbf{v}, \mathbf{h}')) = 1.$$

References

[1] L. J. P. van der Maaten, "t-SNE", `https://lvdmaaten.github.io/tsne`.

[2] L. J. P. van der Maaten and G.E. Hinton, Journal of Machine Learning Research **9**: 2579-2605 (2008).

[3] J. Carrasquilla and R. G. Melko, Nat. Phys. **13**, 431 (2017), `https://arxiv.org/abs/1605.01735`.