Computational Statistical Physics

Part I: Statistical Physics and Phase Transitions

Marina Marinkovic

March 30, 2022

ETH Zürich
Institute for Theoretical Physics
HIT G 41.5
Wolfgang-Pauli-Strasse 27
8093 Zürich

402-0812-00L
FS 2022

Machine Learning Classification

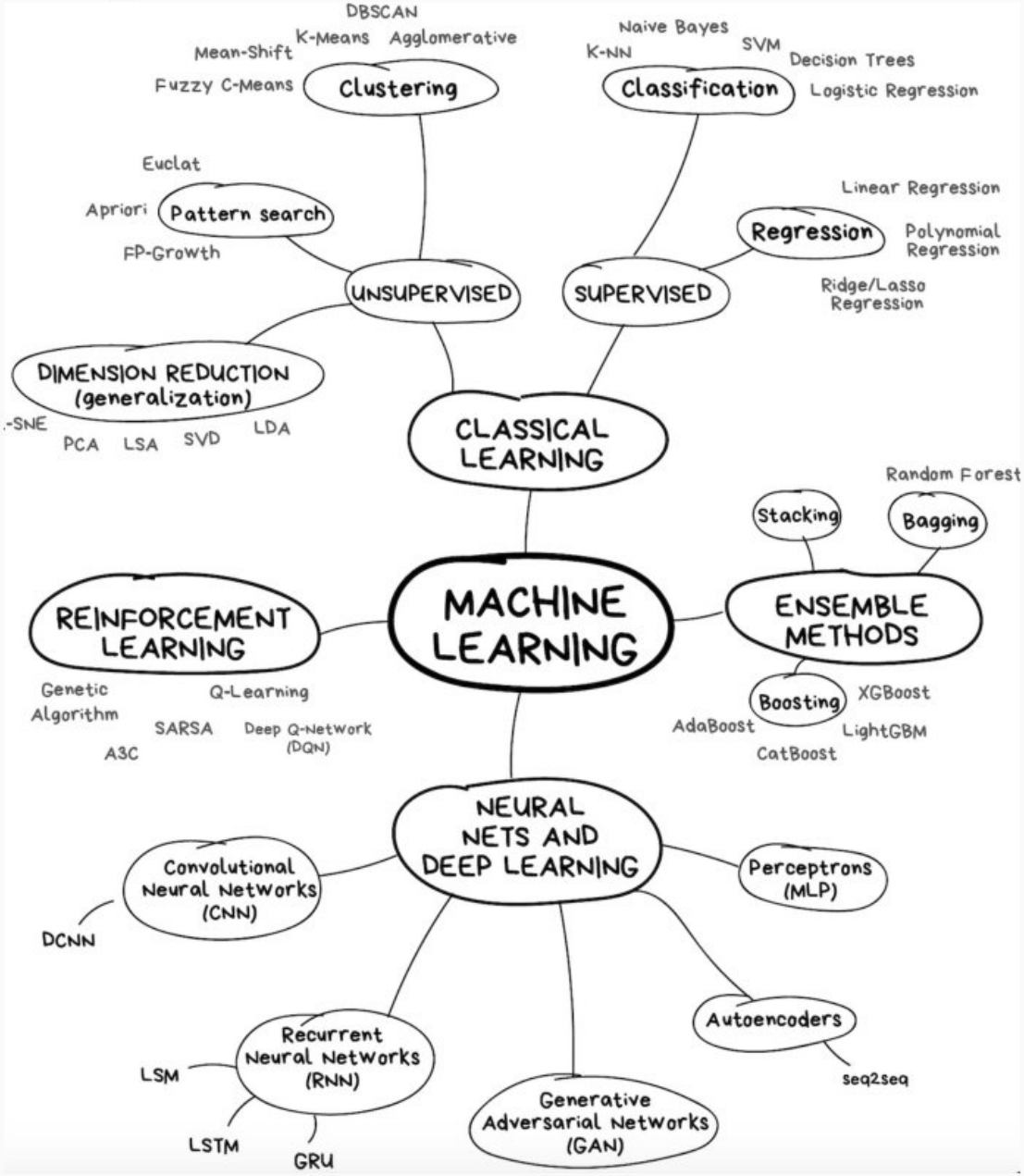

Figure 1: [Credit: https://vas3k.com/blog/machine_learning/]

Boltzmann machine

Hopfield Network

We begin our excursion to machine learning with a network consisting of neurons which are fully connected, i.e., every single neuron is connected to all other neurons. A neuron represents a node of network and is nothing but a function of I different inputs $\{x_i\}_{i\in\{1,\dots,I\}}$ which are weighted by $\{w_i\}_{i\in\{1,\dots,I\}}$ to compute and output y.

Hopfield Network

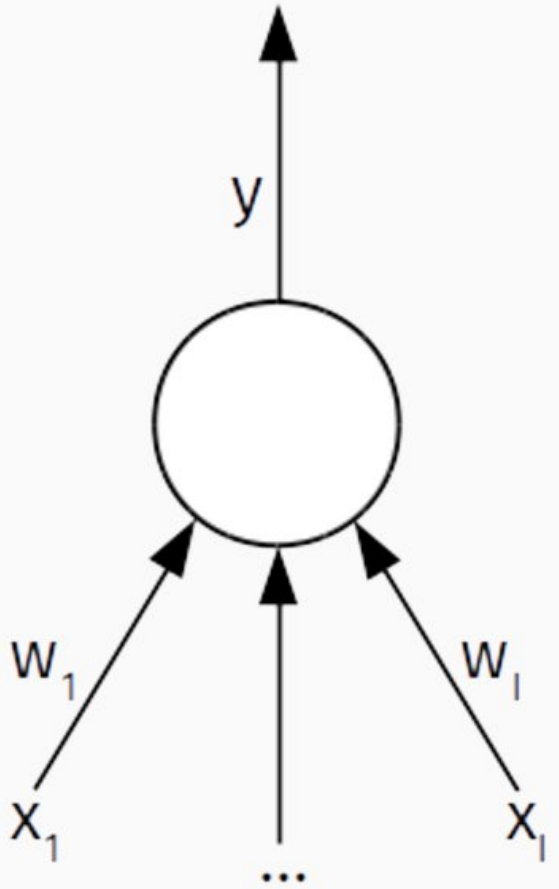

Figure 2: An illustration of a single neuron with output y, inputs $\{x_i\}_{i\in\{1,\ldots,I\}}$ and weigths $\{w_i\}_{i\in\{1,\ldots,I\}}$.

Hopfield Network

In terms of a Hopfield network, we consider discrete inputs $x_i \in \{-1, 1\}$. The activation function of neuron i is given by

$$a_i = \sum_j w_{ij} a_j, \tag{1}$$

where we sum over the inputs. The weights fulfill $w_{ij} = w_{ji}$ and $w_{ii} = 0$.

Hopfield Network

Similarly to the Ising model, the associated energy is given by

$$E = -\frac{1}{2}\sum_{i,j} w_{ij} x_i x_j - \sum_i b_i x_i, \tag{2}$$

where b_i is bias term.

Hopfield Network

The dynamics of a Hopfield network is

$$x_i(a_i) = \begin{cases} 1 & \text{if } a_i \geq 0, \\ -1 & \text{otherwise.} \end{cases} \tag{3}$$

Hopfield Network

The energy difference ΔE_i after neuron i has been updated is

$$\Delta E_i = E(x_i = -1) - E(x_i = 1) = 2\left(b_i + \sum_j w_{ij} x_j\right) \qquad (4)$$

We can absorb the bias b_i in the sum by having an extra active unit at every node in the network. We thus showed that the activation defined by Eq. (1) amounts to the half of the energy difference ΔE_i.

Boltzmann machine learning

Figure 3: A comparison of denoising capabilities of Hopfield and Boltzmann machine models. The figure is taken from `https://arxiv.org/pdf/1803.08823.pdf`.

Boltzmann machine learning

For some applications finding a local minimum based on the deterministic update rule defined by Eq. (3) might not be sufficient. Similar to the discussion of Monte Carlo methods for Ising systems, we employ an update probability

$$p_i = \frac{1}{1 + \exp(-\Delta E_i / T)} = \sigma(2a_i/T) \tag{5}$$

to set neuron i to unity independent of its state [Ackley et al., 1985]. Here, $\sigma(x) = 1/[1 + e^{-x}]$ denotes the sigmoid function.

Boltzmann machine learning

As defined in Eq. (4), the energy difference ΔE_i is the gap between a configuration with an active neuron i and an inactive one. The parameter T acts as temperature equivalent[1].

A closer look at Eqs. (2) and (5) tells us that we are simulating a Hamiltonian system with Glauber dynamics. Due to the fulfilled detailed balance condition, we reach thermal equilibrium and find again for the probabilities of the system to be in state X or Y[2]

$$\frac{p_{\text{eq}}(Y)}{p_{\text{eq}}(X)} = \exp\left(-\frac{E(Y) - E(X)}{T}\right). \tag{6}$$

[1] For $T \to 0$, we recover deterministic dynamics as described by Eq. (3).
[2] Here set $k_B = 1$.

Boltzmann machine learning

We divide the Boltzmann machine units into visible and hidden units represented by the nonempty set V and possibly empty set H, respectively. The visible units are set by the environment whereas the hidden units are additional variables which might be necessary to model certain outputs.

Boltzmann machine learning

Let $P'(\nu)$ be the probability distribution over the visible units ν in a freely running network. It can be obtained by marginalizing over the corresponding joint probability distribution, i.e.,

$$P'(\nu) = \sum_h P(\nu, h) \tag{7}$$

where h represents a hidden unit.

Boltzmann machine learning

The goal is to come up with a method such that $P'(\nu)$ approaches the unknown environment distribution $P(\nu)$. We measure the difference between $P'(\nu)$ and $P(\nu)$ in terms of the Kullback-Leibler divergence (relative entropy)

$$G = \sum_{\nu \in V} P(\nu) \ln \left[\frac{P(\nu)}{P'(\nu)} \right]. \tag{8}$$

Boltzmann machine learning

To minimize G for the freely running network, we perform a gradient descent according to

$$\frac{\partial G}{\partial w_{ij}} = -\frac{1}{T}\left(p_{ij} - p'_{ij}\right), \tag{9}$$

where p_{ij} is the probability that two units are active on average if the environment is determining the states and p'_{ij} is the corresponding probability in a freely running network without coupling to the environment.

Boltzmann machine learning

Both probabilities are measured at thermal equilibrium. In the literature, the probabilities p_{ij} and p'_{ij} are also often defined in terms of the thermal averages $\langle x_i x_j \rangle_{\text{data}}$ data and $\langle x_i x_j \rangle_{\text{model}}$ model, respectively. The weights wij of the network are then updated according to

$$\Delta w_{ij} = \epsilon \left(p_{ij} - p'_{ij} \right) = \epsilon \big(\langle x_i x_j \rangle_{\text{data}} - \langle x_i x_j \rangle_{\text{model}} \big), \qquad (10)$$

where ϵ is the learning rate.

Boltzmann machine learning

The so-called restricted Boltzmann machine is not taking into account mutual connections within the set of hidden and visible units, and turned out to be more suitable for learning tasks. In the case of restricted Boltzmann machines, the weight update is given by

$$\Delta w_{ij} = \epsilon\big(\langle \nu_i h_j \rangle_{\text{data}} - \langle \nu_i h_j \rangle_{\text{model}}\big), \tag{11}$$

where ν_i and h_j represent visible and hidden units, respectively.

Boltzmann machine learning

Instead of sampling the configurations for computing $\langle \nu_i h_j \rangle_{\text{data}}$ and $\langle \nu_i h_j \rangle_{\text{model}}$ at thermal equilibrium, we could also just consider a few relaxation steps. This method is called contrastive divergence and defined by the following update rule

$$\Delta w_{ij}^{CD} = \epsilon \big(\langle \nu_i h_j \rangle_{\text{data}}^{0} - \langle \nu_i h_j \rangle_{\text{model}}^{k} \big), \tag{12}$$

where the superscript indices indicate the number of updates.

Parallelization

Monte Carlo parallelization

Monte Carlo simulations on regular lattices are well-suited for parallelization. The following points summarize the basic ideas behind parallelization:

Domain decomposition

- Nearest-neighbor updates do not involve the whole system.
- Domain decomposition into sublattices is possible.
- The decomposed domains are distributed using MPI.
- Sublattices are extracted with logical masks.
- A periodic shift (CSHIFT) is used to obtain neighbors for periodic boundary conditions.