Course mini-project

The aim of the present project is to code and to experiment the Markov Chain Monte-Carlo (MCMC) method for the so-called perceptron. One component of the project is to get familiar with the idea of *simulated annealing*, where the temperature parameter is lowered progressively. This is an open project were you have to experiment in order to improve the performance of the method.

The perceptron

Consider the following learning problem. One has a set of M "patterns" modelled as N-dimensional vectors $\underline{x}_\mu \in \mathbb{R}^N$, $\mu = 1, \ldots, M$. Each pattern belongs to one of two "classes" $y_\mu \in \{-1, +1\}$. One would like to "learn" a vector of "weights" $\underline{w}$ such that the following constraints are satisfied (or at least the number of errors is minimized):

$$y_\mu = \mathrm{sign}(\underline{w} \cdot \underline{x}_\mu), \quad \mu = 1, \ldots, M$$

Geometrically, $\underline{w}$ corresponds to a vector perpendicular to a hyperplane that separates/classifies the patterns into two classes. The aim is to minimize the total cost or energy function, defined as

$$E(\underline{w}) = \frac{1}{2} \sum_{\mu=1}^{M} (y_\mu - \mathrm{sign}(\underline{w} \cdot \underline{x}_\mu))^2$$

which counts the number of misclassifications for a vector $\underline{w}$.

For $\underline{w} \in \mathbb{R}^N$, this model is called the binary perceptron (because there are two classes) and for $\underline{w} \in \{-1, +1\}^N$, it is called the Ising perceptron because the weights can be seen as Ising spins and $E(\underline{w})$ as an "Ising type" Hamiltonian or energy function. One is interested in particular in the regime where the ratio $\alpha = M/N$ is fixed and N, M are both large (in theory, $N, M \to +\infty$).

The teacher-student scenario

In the teacher-student scenario, a "teacher" has a separating plane $\underline{w}^*$ and generates a set of patterns $\underline{x}_\mu$, as well as their class labels $y_\mu = \mathrm{sign}(\underline{w} \cdot \underline{x}_\mu)$. The "student" has access to the pairs $(y_\mu, \underline{x}_\mu)$ called the "training set" and her task is to find $\underline{w}^*$. We assume that $\underline{w}^* \in \{-1, +1\}^N$ has i.i.d. Ber(1/2) components and that $\underline{x}_\mu$ have i.i.d. Gaussian components with mean zero and unit variance. Of course, in this case, $\underline{w}^*$ is a minimizer of $E(\underline{w})$ and $E(\underline{w}^*) = 0$. One expects that for α large enough, the student should be able to learn $\underline{w}^*$, while if α is too small, then she might find a wrong solution.

MCMC method

The student decides to set up an MCMC chain by looking at a finite temperature version of the problem. Consider the *Gibbs-Boltzmann* probability distribution

$$p(\underline{w}) = \frac{1}{Z} e^{-\beta E(\underline{w})}, \qquad Z = \sum_{\underline{w} \in \{-1,+1\}^N} e^{-\beta E(\underline{w})}$$

For the moment, think of $\beta > 0$ fixed. We design the following Metropolis chain:

1. Start from a random initial vector $\underline{w}^{(0)} \in S$;
2. The base chain has the following transition mechanism: at each step, choose a coordinate i of the current vector $\underline{w}$ uniformly at random and flip it, so that $\underline{w}' = (w_1, \ldots, w_{i-1}, -w_i, w_{i+1}, \ldots, w_N)$. Accept then the move with probability

$$a_\beta(\underline{w}, \underline{w}') = \min(1, \exp(-\beta(E(\underline{w}') - E(\underline{w}))))$$

3. Iterate point 2 to obtain the Markov chain $(\underline{w}^{(t)}, t = 0, \ldots, T)$ until the energy $E(\underline{w}^{(T)})$ is sufficiently low.

All this with the hope that a state $\underline{w}^{(T)}$ of low energy is as close as possible to the teacher's vector $\underline{w}^*$ when β is large enough. In this project, we first ask you to perform the following tasks:

1. For a fixed value of α and β, run the algorithm and draw the energy $E(\underline{w}^{(t)})$ as a function of t. For each fixed value of α and β, you may run the algorithm multiple times and compute the averaged energy in order to get better curves. Repeat the operation for multiple values of α and β.
2. Fix now the value of β, choose T sufficiently large, and draw the *normalized* energy $e(\alpha, \beta) = \frac{1}{M} E(\underline{w}^{(T)})$ reached at time T by the algorithm as a function of α. Again, it is a good idea to average over multiple runs of the algorithm in order to get nicer curves. Repeat the operation for multiple values of β.
3. Similary, fix the value of β and draw the *overlap* $q(\alpha, \beta) = \frac{1}{N} \sum_{i=1}^{N} w_i^{(T)} w_i^*$ as function of α. Repeat the operation for multiple values of β. Again, average over multiple runs of the algorithm in order to get nicer curves. The value of the overlap q lies always in the interval $[-1, +1]$. It gives an indication of how close the vector $\underline{w}^{(T)}$ found by the algorithm is from the ground truth $\underline{w}^*$ (note that $q = +1$ if and only if $\underline{w}^{(T)} = \underline{w}^*$).

Important remarks

- The values of α and β where you should see something interesting happening are typically located in the interval $[0.5, 5]$ for both variables. Please do not draw uncountably many graphs, but just the ones for which one observes significant differences.
- The value of N is left to your own appreciation; it should be as large as possible, but of course also within the limits of the computing power of your machine.
- Please be aware that while varying N and M with a fixed ratio α, the number of steps T performed by the algorithm should be scaled linearly with N.

Simulated annealing

Your second task in this project is to use simulated annealing to optimize the performance of your algorithm. For a given value of α, you are free to choose 1) the initial value of β, 2) at which pace you increase the value of β and 3) by how much you increase the value of β at each step, in order to get the lowest possible energy and best possible overlap at the end of the algorithm. Draw again the (averaged) normalized energy $\frac{1}{M} E(\omega^{(T)})$ and the overlap $\frac{1}{N} \sum_{i=1}^{N} w_i^{(T)} w_i^*$ reached by your algorithm, as a function of α.

Deadline

You should provide us a short report (2-4 pages) along with your code (both sent by email to Clement, Nicolas and Olivier) by Friday, December 15. The code can be written in the language you prefer. Typically, Matlab or Python will do.

Beyond the teacher-student scenario

We will tell you more about this during the final competition on Thursday, December 21, at 12:15 PM. Be prepared!