Course mini-project

The aim of the present project is to code and to experiment the Markov Chain Monte-Carlo (MCMC) method for a generalized linear estimation problem. 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.

Non-linear estimation problem

Let $W = \{W_{ij}\}$ be a matrix of dimension $m \times n$ with entries $W_{ij} \overset{iid}{\sim} \mathcal{N}(0,1)$ for $1 \le i \le m$, $1 \le j \le n$. This matrix is *known* and one would like to recover a vector $X = (X_1, \dots, X_n)^T \in S = \{-1,1\}^n$ from the following m observations:

$$Y_i = \varphi\left(\sum_{j=1}^{n} \frac{W_{ij}}{\sqrt{n}} X_j\right) = \varphi\left(\left[\frac{WX}{\sqrt{n}}\right]_i\right), \; 1 \le i \le m\,.$$

Here $\varphi : x \mapsto \max\{0,x\}$ is the ReLU – for *Rectifier Linear Unit* – function. The m observations can be summed up into a vector $Y = \varphi\left(\frac{WX}{\sqrt{n}}\right)$ (φ acts component-wise when applied to a vector).

As the problem is non-linear and the search space is large (of size 2^n), it is not clear how to solve this problem efficiently. We propose here to approach the problem via the MCMC method.

For a given observation Y, we define the *energy* of a vector $x \in S = \{-1,+1\}^n$ as

$$H_Y(x) = \sum_{i=1}^{m} \left| Y_i - \varphi\left(\left[\frac{Wx}{\sqrt{n}}\right]_i\right)\right|^2 \tag{1}$$

Our aim is to minimize this energy via the MCMC method. For this purpose, we introduce an auxiliary *finite temperature* version of the problem. Consider the *Gibbs-Boltzmann* probability distribution

$$p_Y(x) = \frac{e^{-\beta H_Y(x)}}{Z_\beta}, \quad x \in S$$

where $\beta = 1/T$ is the *inverse temperature* and $Z_\beta = \sum_{x\in S} e^{-\beta H_Y(x)}$ is the normalizing factor (also called the *partition function*). The idea described below is to construct a Markov chain that samples correctly from this distribution. As we take the parameter $\beta \to +\infty$ (equivalently $T \to 0$), the obtained sample should converge to a vector x that minimizes the above energy function (1).

Metropolis chain

For the moment, think of $\beta > 0$ fixed. For this, we design the following Metropolis chain on the state space S:

1. Start from a random initial vector $x^{(0)} \in S$.
2. The base chain has the following transition mechanism: at time t, choose a coordinate i of the current vector $x^{(t)} \in S$ uniformly at random and flip the sign of the corresponding entry, so that
$$x^{(t+1)} = (x_1^{(t)}, \dots, x_{i-1}^{(t)}, -x_i^{(t)}, x_{i+1}^{(t)}, \dots, x_n^{(t)})\,.$$
NB: Does this base chain satisfy the assumptions of the theorem seen in class?
3. Accept the previous move with probability
$$a_\beta(x^{(t)}, x^{(t+1)}) = \min\left\{1, e^{-\beta(H_Y(x^{(t+1)}) - H_Y(x^{(t)}))}\right\}.$$
If the move is rejected then $x^{(t+1)} = x^{(t)}$.
4. Iterate point 2 to obtain the Markov chain $x^{(t)}$, $t = 0, \dots, t_{\max}$ until the energy $H_Y(x^{(t_{\max})})$ is sufficiently low.

All this with the hope that a state x of low energy $H_Y(x)$ is as close as possible to the original vector X.

An alternate algorithm: Glauber or heat bath dynamics

1. Start from a random initial vector $x^{(0)} \in S$.
2. At time t select a coordinate i at random from $\{1, \cdots, n\}$, erase the value of $x_i^{(t)}$, and reset this value to ± 1 with probabilities
$$p_{\pm} = \frac{1 \pm x_i^{(t)} \tanh\left(\beta\left(H_Y\left(\tilde{x}^{(t)}\right) - H_Y\left(x^{(t)}\right)\right)\right)}{2}, \quad \tilde{x}_i^{(t)} = -x_i^{(t)}, \ \forall j \neq i : \tilde{x}_j^{(t)} = x_j^{(t)}.$$
This yields a new vector $x^{(t+1)}$ (the derivation of this formula is exactly the same as for the Ising model seen in class; try it explicitly!).
3. Iterate point 2 to obtain the Markov chain $x^{(t)}$, $t = 0, \ldots, t_{\max}$ until the energy $H_Y(x^{(t_{\max})})$ is sufficiently low.

You are free to choose the algorithm (Metropolis or Glauber) for your implementation. An even better option is to try both and opt for the best!

Simulated annealing

The main idea in simulated annealing is to lower the temperature $T = \frac{1}{\beta}$ as time goes by. Note that the transition probabilities change with time now, so the Metropolis chain is not anymore time-homogeneous.

You will have to experiment to find a suitable schedule for lowering T (resp., increasing β). If you lower T too slowly, you might not reach a minimizer of $H_Y(x)$ in a decent amount of time. If you lower T too quickly, you lose the advantage of the Metropolis step and might end up in a local minimum with a high energy.

One (basic) advice is not to change the temperature at every iteration step, but rather to keep it constant for some steps before lowering the temperature.

Reconstruction error

Let $\hat{x}$ be your estimate for the ground truth vector X. We define the following reconstruction error:
$$e(\hat{x}, X) = \frac{1}{4n}\left\|\hat{x} - X\right\|^2,$$
with $\|\cdot\|$ the Euclidian norm.

Question: Show in your report that $e(\hat{x}, X)$ is the fraction of entries on which $\hat{x}$ differs from X.

What we expect from you

Below the guideline for this project:

1. Form a team of 3 students to complete this project and choose a team name. One team member should send an email to `clement.luneau[at]epfl.ch`, `olivier.leveque[at]epfl.ch`, `nicolas.macris[at]epfl.ch`, with the details of your team, by **Wednesday, November 29**.
2. Implement a version of the above MCMC method, along with simulated annealing, which finds, for a given observation Y, a vector $\hat{x}$ with error $e(\hat{x}, X)$ as low as possible.
3. Describe and optimize your "cooling strategy" for simulated annealing. Plot the error $e(\hat{x}, X)$ obtained by your algorithm as a function of time.
4. Run multiple experiments in order to estimate
   - the expected error $\mathbb{E}[e(\hat{x}, X)]$;
   - the standard deviation of the error $\sqrt{\mathbb{V}\text{ar}[e(\hat{x}, X)]}$;

reached by your algorithm. By experiment, we mean the process of drawing W and X, forming the observation Y and running the algorithm to estimate X. Plot the result as a function of the parameter $\alpha = \frac{m}{n}$ (α is the ratio of the number of observations to the number of unknowns). Choose the range of α wisely, so as to see an interesting behavior (see following point).

5. Do you observe a critical value $\alpha_c > 0$ corresponding to an abrupt change in your algorithm's performance? Is there a limiting value $\alpha_{\mathrm{rdm}} > 0$ below which your algorithm does not perform better than a purely random guess?

The code can be written in the language you prefer. Typically, Matlab or Python will do. Try to optimize your algorithm: the larger the value of n, the better! Especially if you want to observe something for the step 5.

You should then

- produce a small report (2-4 pages) answering the above questions, as well as a short description of your code **(due Tuesday, December 11)**;
- be prepared to participate to the final competition on **Thursday, December 20, at 12:15 PM**!

Deadline

You have to submit your report along with your code on Moodle **before Tuesday, December 11, at 11:59 PM**.