Assignment #2: Application of Sampling

Vladimir Ivanov, Zamira Kholmatova

April 2024

Due Date: May 5, 23:59
Evaluation Form: IPython notebook

You are **NOT** allowed to use ChatGPT or any other LLMs to accomplish this assignment. All students suspected in the use of LLMs will be reported to DoE for plagiarism.

The goal for this homework is to implement an optimization algorithm called Simulated Annealing (SA). It is based on sampling and the optimization procedure is similar to the Metropolis-Hastings (MH) algorithm. Because the optimization is based on sampling, SA allows to optimize even non-differentiable functions, and thus it is applicable to optimizing combinatorial problems.

Motivation

SA is a stochastic global optimization algorithm. As many other randomized algorithms, SA appears to be surprisingly simple, yet efficient at achieving its task (given conditions). The name and inspiration for the algorithm come from metallurgy, where controlled heating and cooling of the material allows it to form larger crystalline structure and improve properties. A popular description of the method can be found on Wikipedia page (`https://en.wikipedia.org/wiki/Simulated_annealing`).

Algorithm

Procedure

The procedure for SA is very similar to the MH algorithm:

1. Sample initial x_0, set time step $t = 0$;
2. Set initial temperature T;

3. Generate x' from $g(x'|x_t)$;
4. Calculate acceptance ratio $\alpha = \frac{p^*(x')}{p^*(x_t)}$;
5. Generate $u \sim U(0,1)$. If $u \leq \alpha$, accept the new state $x_{t+1} = x'$, otherwise propagate the old state. Pass x_{t+1} to the output of the sampler;
6. Anneal temperature T;
7. Increment time step t;
8. Repeat 2-8 until cooled down.

The description and details of every step are given further.

Metallurgy Analogy

When the solid is heated, its molecules start moving randomly, and its energy increases. If the subsequent process of cooling is slow, the energy of the solid decreases slowly, but there are also random increases in the energy governed by the Boltzmann distribution. If the cooling is slow enough and deep enough to unstress the solid, the system will eventually settle down to the lowest energy state where all the molecules are arranged to have minimal potential energy.

The terms in SA are borrowed from the field of metallurgy. The optimization problem (minimization or maximization) is treated as a system. We want it to achieve a stable state with the least of energy (determined by cost function). This is achieved by heating the system up (increasing the variance of sampling), and then gradually cooling it down. This metallurgical process is merely an abstraction, and the actual optimization relies on sampling from an artificially designed probability distribution that peaks around global optimum. The way to construct this probability distribution is described further.

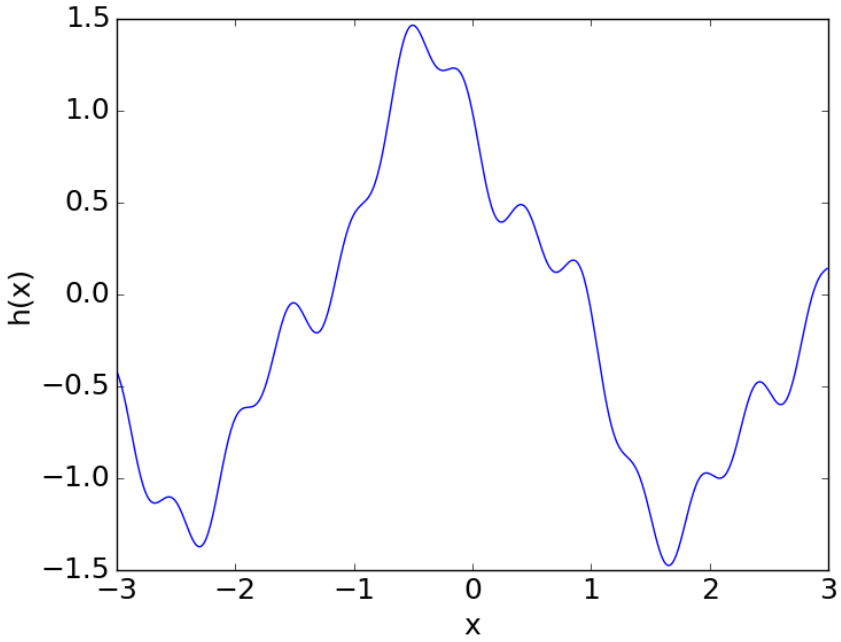
Energy Distribution

The main trick of SA is the energy distribution. This distribution peaks around the point of the global optimum, and thus, there is a good chance to sample this value. First, consider how the energy distribution is constructed.

SA steps are similar to the Metropolis-Hastings algorithm. Remember that the idea behind MS algorithm is to create an MCMC that samples from the distribution proportional to $f(x)$, so that the limiting distribution is equal to the target distribution. The idea of Simulated Annealing method is similar.

Assume you have a function $h(x)$ that you want to minimize in the region $x \in [3,3]$. In simulated annealing, we say that we have a system, and the function $h(x)$ describes its state.

```
w = np.array([0.1351, 0.3056, 0.8578, 0.1984, 0.0972])
phi = np.array([0.7663, 0.0761, 0.5743, 2.0537, 1.9737])
a = np.array([12.8047, 1.5739, 1.4613, 4.9329, 4.1203])
h = np.vectorize(lambda x: sum(np.cos(x*a+phi)*w))
```



As you can see, the non-concavity of the function makes it hard to optimize with gradient descent. Define the cost function, also referred to as system energy

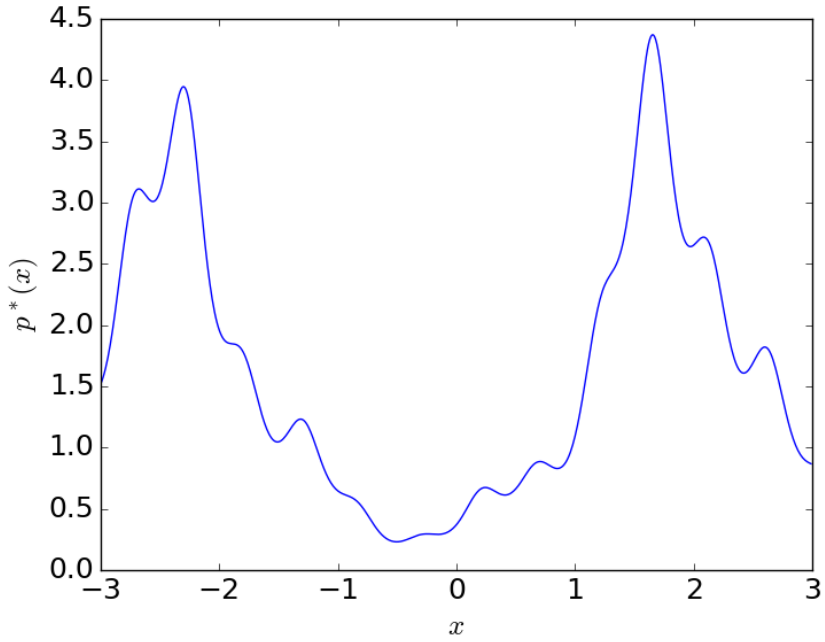
$$E(x) = h(x)$$

for maximization problems the sign of $h(x)$ would be negative. This function is used to define the energy distribution. Our goal is to create the distribution, where the global optimum has a high weight. One of the possible transformations adopted by SA is

$$p(x) = \frac{e^{-E(x)}}{\int e^{-E(y)} dy}$$

This transformation is inspired by Boltzman Distribution (`https://en.wikipedia.org/wiki/Boltzmann_distribution`) and is similar to softmax. We do not want to compute the denominator of this distribution since it would imply passing through the entire search space (if we do this, we can find minimum right away).

Notice that the density maximum is observed exactly where the global minimum is. The problem is, even if we sample from this distribution, there is no guarantee the sample will be close to the desired minimum. This issue is solved with annealing.



Annealing

Annealing is the process of cooling down. When the system is heated, we want to emulate Brownian motion by transitioning the state often. In other words, when the system is heated there should be a high chance for the system to change its state. From a sampling perspective, this is achieved by making the peaks in the distribution less distinct. In SA, this is done by adding temperature parameter to the target distribution function as follows

$$p(x) = \frac{e^{-E(x)/T}}{\int e^{-E(y)/T} dy}$$

And the proportional function is

$$p^* = e^{-E(x)/T}$$

By gradually reducing the temperature T we make the peaks more pronounced, and encourage the system to stabilize in the point with the minimal energy. In our example, we end up sampling from the distribution with only one peak.

Annealing Schedule

Throughout the optimization procedure, the temperature is gradually annealed. The annealing schedule specifies the speed of the temperature decrease. The

most common way to anneal is with exponential decay (`https://en.wikipedia.org/wiki/Exponential_decay`) or a geometric series

$$T_{t+1} = \gamma T_t$$

where $\gamma < 1$ is the annealing rate. From MH algorithm we know that the sampling procedure has the burn-in time. If the temperature is decreased too fast, the process may end up stuck in a local optimum.

Proposal Distribution

The problem at hand determines the choice of the proposal distribution. For continuous space problems, the Normal proposal distribution is a common choice. The alternative is described in the further section about combinatorial optimization.

Putting Everything Together

The optimization procedure is similar to the MH algorithm

1. Sample initial x_0, set time step $t = 0$;
2. Set initial temperature T. To avoid problems with numerical overflow when calculating the exponent, set the temperature comparable with the initial value of the system's energy;
3. Generate x' from $g(x'|x_t)$. For continuous problems, the common solution is to use the normal distribution;
4. Calculate acceptance ratio $\alpha = \frac{p^*(x')}{p^*(x_t)}$;
5. Generate $u \sim U(0, 1)$. If $u \leq \alpha$, accept the new state $x_{t+1} = x'$, otherwise propagate the old state. Pass x_{t+1} to the output of the sampler;
6. Reduce temperature T. Temperature annealing does not have to take place on every iteration. The temperature can be decreased every N iteration as well. There is no strict rule about this;
7. Increment t;
8. Repeat 2-8 until cooled down. The solution can be sampled before the system is cooled down, but we have no way of knowing whether this was the optimal solution

Combinatorial optimization

Due to the nature of this optimization approach, it does not require the function to be differentiable. This allows to apply this method to optimize combinatorial problems.

Consider the traveling salesman problem

Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city once and returns to the origin city?

The energy distribution for this problem is easy to express in a format suitable for SA.

$$p^*_{\text{sales}}(\text{path}) = \exp \frac{-\text{dist}(\text{path})}{T}$$

where *path* is the ordered list of cities to visit, *dist* is the function that computes path distance. The main trick to solve traveling salesman with SA is to pick a proper proposal distribution. The common proposal policy for this problem is the following:

1. Pick two cities in the path;
2. Exchange their positions in the path;
3. Return the new proposed path.

With the energy and proposal distribution defined as such, the optimization process will look the following way – see video (`https://youtu.be/pXEzDMz0jnc`).

Task

1. Download the location data for cities in Russia - `https://github.com/hflabs/city`;
2. Find the optimal traveling salesman path using SA for 30 most populated cities (*remember that the dataset contains geodesic coordinates, calculating the distance would require to apply a suitable transformation*);
3. For your selected annealing schedule, track the speed of convergence for three different values of the annealing rate (*try fast cooling, slow cooling, and some middle value*). Compare the optimization result;
4. Upload your notebook with **all cells executed** to Moodle.

References

1. Simulated Annealing - `http://www.mit.edu/~dbertsim/papers/Optimization/Simulated20annealing.pdf`

2. Travelling Salesman with SA (https://www.hindawi.com/journals/cin/2016/1712630/, https://www.fourmilab.ch/documents/travelling/anneal/, https://toddwschneider.com/posts/traveling-salesman-with-simulated-annealing-r-and-shi //www.theprojectspot.com/tutorial-post/simulated-annealing-algorithm-for-beginners/ 6)