Learning From Randomness

by Sven Nilsen, 2017

Given any labyrinth and randomly moving particles, the distribution of an exponentially growing number of particles starting in the same state has the property that for any state, most particles take the shortest path. This happens due to a selection effect, since most particles recently started out in the initial state. This can be translated into a theorem that the gradient of neighbor states points toward the source, so by learning the distribution and then tracing the path back using the gradient, one can derive the shortest path. The surprising property of a such algorithm is that learning happens by randomly exploring the space, without seemingly any form of improvement in efficiency, until the final collected data yields the optimal solution. I demonstrate that this algorithm also works on global optimization problems.

In the Traveling Salesman Problem (TSP), a salesman need to find the shortest route through N cities while visiting all of them. This is one of the most well studied problems in optimization theory and already have very efficient heuristic algorithms. It is also used as a benchmark for optimization algorithms. The algorithm I will describe performs much worse than the fastest algorithms, but it has some fascinating properties that becomes clear when using it to solve TSP.

The algorithm is as following:

1. Choose a constant α between 0 and 1
2. Create an NxN matrix to store weights
3. Generate a random route between the cities
4. Calculate the length of the route L
5. Compute α^L and add it to the weight matrix for each edge pair of cities in the random route
6. Repeat step 2 to 5 as many times as possible up to some time limit
7. Trace the maximum values in the weight matrix to find the approximately shortest route

This algorithm is easy to implement and trivial to parallelize,
yet it has some very strange and beautiful properties:

- You can pause the training and continue later by storing the weight matrix in a file
- Two separate training sessions can be combined by adding the weight matrices together
- Experience is learned from random routes only

The algorithm is derived as an approximation to the distribution of exponentially growing, randomly moving particles in a labyrinth. Instead of measuring all possible states, it is measuring only the states of particles that complete the route. As long as the number of particles are exponentially growing, the conditions we use for the state does not matter. The constant α is the inverse of the exponential growth rate `k` per unit of distance:

$$\alpha = 1 / k$$

Intuitively, one can imagine that a random route gets coincidentally short because of some traveling pattern between a subset of the cities. This adds up in the weight matrix as if this route was travelled

many times, compared to longer routes. The lower α, the higher number of particles traveling along shorter routes. If the shortest route is found by random, it gets the highest boost in the weight matrix.

In theory it does not matter what value α has on continuous distances, as long it is between 0 and 1, because the direction of the gradient will be the same. However, a too low value can lead to problems with numerical accuracy and making it biased toward some short routes. Setting it closer to 1 means random routes can learn more from each other.