Milan Zanussi

Introduction to Artificial Intelligence

October 25, 2018

A Comparison of Greedy Local Search to Simulated Annealing

Optimization is perhaps one of the most significant forms of applied mathematics, and its ubiquitous utility in various fields of study ranging from business and economics to the natural sciences has made the practice itself a field of specialization for many mathematicians. Centuries of research have been dedicated to solving optimization problems whenever they have appeared in all of these various fields of study, and have inspired the development of powerful mathematical tools, such as the development of basic integration by Archimedes to optimize the buoyancy of hulls of boats. Another optimization development came from studying the principle of least action in classical mechanics, which eventually led to the birth of the calculus of variations [2].

Artificial intelligence stands among the numerous fields which rely immensely on the use of optimization for solving many of the problems within the field. Often artificial intelligence researchers will categorize environments into discrete and continuous environments – continuous meaning that the environment is a continuum, which one can intuitively imagine as being a space where between any two points, at least one other distinct point exists between them such as the real number line; and discrete meaning the space is not continuous, such as the integers where there is no number between any two consecutive integers. The goal of optimization over either kind of domain is to apply some function over the domain which an agent can compare its possible choices – usually ascribing real number values to the possible choices – and then choosing whichever choice seems optimal. The function which assigns comparative values to each point in the environment serves as a way of measuring the performance of an artificial agent in the given environment.

In general, one method which is useful for optimization is the use of comparative difference measurements to decide which changes in inputs lead to a more optimal output. In a discrete environment, these methods are called finite difference methods, and they are as simple as taking differences between function outputs when one parameter has been perturbed by some amount. The analogue to this method in continuous spaces which are endowed with a sufficiently smooth and connected structure is called differentiation, which allows us to take sufficiently smooth functions, which we call differentiable functions, and find derived functions which tell us how the function is changing instantaneously at a given point, which is called the derivative. Sometimes the difference methods used with discrete methods can be used with non-differentiable functions for optimization, however they often fail to provide the same level of effectiveness as the derivative methods.

We may also consider whenever derivative functions are themselves differentiable, and their derivatives are called the higher-order derivatives of the function from which they were derived. In higher dimensions, functions are allowed to be differentiable along any path approaching particular points. Whenever derivatives are taken on real-valued functions with respect to individual variables upon which the function depends, we call these derivatives partial derivatives, as they only consider differentiation in one direction. If we take derivatives along directions which vary multiple parameters simultaneously, we call them directional derivatives. These directional derivatives describe how the functions changes instantaneously along the particular directions in which the functions are differentiated, and so a local search strategy for optimal solutions can be formulated by searching for the directions of most change.

One important theorem regarding multivariable derivatives regards a construct called the gradient. The gradient is a vector function, or a function with multiple outputs which are each called the components of the vector, whose components are each of the partial derivatives of the function with respect to each of its parameters. The theorem of significance tells us that the gradient always points in the direction of the highest increase for a differentiable function, and the negation of the gradient points in the direction of highest decrease at the given point. This fact relating the directions of maximum change to the partial derivatives becomes the basis for most of the theory of local search over continuous spaces.

Our first algorithm used in this study uses this theorem to find maximal points by following the gradient along small increments until a maximal point is found. This method is typically called gradient ascent or greedy local search. The method starts at a random initial starting point and will climb to the nearest local maximum on the function [1], however this does not guarantee that the found point is the global maximum of the function over the domain which the function is defined. The greedy local search algorithm will only guarantee optimality on convex functions. A brief mathematical definition of a convex function over real numbers is a function f where for any pair of real numbers a, b and for any real number λ such that $0 < \lambda < 1$ the following inequality holds [3]:

$$f(\,\lambda a + (1-\lambda)b\,) \leq \lambda f(a) + (1-\lambda)f(b)$$

In situations where our functions are not convex, we can no longer guarantee that greedy local search will converge to a global maximum, since it could be possible for there to be multiple local maxima on our function where greedy local search will consider all other directions less optimal at any of the local maxima. However, we can introduce new methods which remove the assumption that the global maximum will be the only point on our function where moving in any direction will reduce the value of the output. The method against which greedy local search will be compared which removes this assumption is called simulated annealing – inspired by the metallurgy technique of annealing where metal is heated to higher temperatures to allow the metal particles to crystallize and recrystallize into different configurations which strengthens the metal.

The principle idea of simulated annealing is to assign a numerical value which describes a degree of randomness which is called the temperature value. The algorithm then randomly perturb the currently known approximation of the optimum of the function

and decide whether or not to change the current approximation to the perturbed approximation based on the temperature and some set of rules [1]. The rules for deciding whether to accept the perturbation varies between implementations. Our study will use the Metropolis criterion, which says that the perturbation will always be accepted if its output value is higher than the known approximation, but if the perturbation gives a lower output, then the algorithm chooses the perturbation with a probability:

$$p(x, \hat{x}) = \exp\left\{\frac{f(\hat{x}) - f(x)}{T}\right\}$$

where f is the function subject to approximation, x is the current approximation of the optimum, $\hat{x}$ is the perturbation of x, and $\exp\{a\}$ simply means to raise e to the power of a for any real number a.

Another aspect of simulated annealing is to choose a temperature schedule, or rather a means at which we assign temperature to current states of the search. The method adopted by this study will be based on the use of another concept from multivariable calculus called the Laplacian operator, and the methodology will be partially inspired by the analytic methodology used to develop the most elementary equation for studying diffusion, which is often called the heat equation due to its history of originating in the study of temperature diffusion. The heat equation is an equation which is solved by a function u depending on time and on some points in space whose derivatives satisfy the following equation:

$$k\nabla^2 u = \frac{\partial u}{\partial t}$$

where $\frac{\partial u}{\partial t}$ is the partial derivative of u with respect to time, $\nabla^2 u$ is the Laplacian operator applied to the function u, and k is a diffusion constant which is usually solved upon assigning some initial value conditions to the function. The Laplacian operator ∇^2 is simply taking the sum of the second-order partial derivatives of u with respect to the assigned spatial dimensions.

The analytic insight which is used to formulate the heat equation is that the Laplacian operator on the spatial dimensions gives a measurement of the local curvature of a function, and furthermore this curvature tends to be near zero in regions of locally similar temperature, positive and large in absolute value when the input maps to a lower point of a function which is near an ascending region of the graph, and negative with a large absolute value when the input maps to a higher point on the function which is near a descending region of the graph. With this in mind, we can formulate a vague sense in which we should be adjusting out temperature. If we are at a low point of the graph where our function is far away from any local maxima, we should increase our temperature so that our search will randomly walk toward any kind of local maxima. Next, we should start to cool down our search when the search approaches the base of some local maxima, so that our function will try to climb the found local maximum. Lastly, once we arrive at the maximum, we want to try to perturb our function to any other neighboring local maxima

which may be higher in value. The temperature function used to try to emulate this behavior in our search was:

$$T_i(x) = \frac{(k-i)}{|\nabla^2 f(x)|}$$

where $T_i(x)$ is the temperature applied to the search on the i-th iteration of the simulated annealing on the function f given the current optimal input approximation is x. In the situation where $|\nabla^2 f(x)| = 0$, we instead let the denominator be equal to 1.

We ran this simulated annealing algorithm and the greedy local search on 100 random sums of Gaussian random variables in 1, 2, 3, 4, and 5 dimensions with 5, 10, 50, 100, 500, and 1000 summands in each of the choices of dimensions and gathered data which produced the following results:

Difference Statistics:	
Minimum:	-2.061039974
Maximum:	3.61181
Mean:	-0.125089294
Median:	0

Optimality Count:		**Weighted:**
Greedy Dominated:	464	21.57455
S.A. Dominated:	406	396.8424316
Neither Dominated:	2130	0

where the difference statistics provide data about the difference $GR(s,d,n) - SA(s,d,n)$ where $SA(s,n,d)$ is the optimum found with simulated annealing using the seed s in d dimensions with n random variables added together, and $GR(s,n,d)$ is the optimum found with greedy local search with the same parameters. The second table counts the number of times each algorithm provided a better optimum than the other, as well as the number of times they provided the same level of optimality. There is also a weighted column, which is simply the amount by which either algorithm dominated in each situation summed together to give a better insight into how much each algorithm dominated whenever it did dominate.

The results of this study would seem to indicate that while the simulated annealing algorithm only marginally outperforms the greedy local search algorithm, as indicated by the mean difference, the greedy algorithm can still sometimes provide a much better solution than the simulated annealing algorithm, as indicated by the maximum of the differences, which indicates that the greedy algorithm had a higher optimum in one situation by an absolute value of about 3.611, we still see that the simulated annealing strategy still has more general gains in many of the situations in which it dominates, and this is mostly due to the robustness of simulated annealing to starting in regions of local flatness, since many higher-dimensional distributions will cause the greedy local search to terminate on the first step, while the simulated annealing will simply wander until it finds a slope to climb.

Works Cited:

[1] Russell, S. J., & Norvig, P. (2014). *Artificial Intelligence: A modern approach*. Harlow, Essex: Pearson.

[2] Britannica, T. E. (2013, August 01). Calculus of variations. Retrieved from https://www.britannica.com/science/calculus-of-variations-mathematics

[3] Rudin, W. (1953). Principles of Mathematical Analysis. New York: McGraw-Hill Book