1 ГРАДИЕНТНАЯ ОПТИМИЗАЦИЯ[1]

Прежде чем перейти к метаэвристикам, рассмотрим традиционный математический метод поиска максимума функции: **градиентный подъем**[2]. Его идея заключается в нахождении наклона функции и движении вверх по нему. Все просто! Будем максимизировать функцию $f(x)$. Данный метод не требует вычисления или даже знания $f(x)$, но он подразумевает, что мы можем посчитать наклон в точке *x*, т.е. что существует $f'(x)$.

Метод очень просто. Начинаем с произвольного значения *x*. Затем последовательно прибавляем к нему небольшую долю наклона, т.е. $\vec{x} \leftarrow \vec{x} + af'(\vec{x})$, где a – малое положительное число. Если наклон положительный, то *x* будет увеличиваться. Если наклон отрицательный, то *x* будет уменьшаться. На рис. 1 показана грубая иллюстрация этого правила.

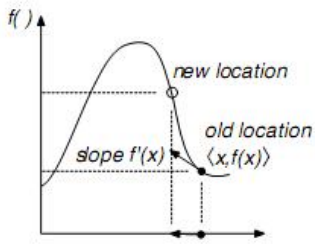

Рис. 1. Градиентный подъем с отрицательным наклоном. *x* уменьшается.

В итоге значение *x* будет двигаться вдоль графика функции, пока не достигнет ее пика, в котором наклон равен нулю и, следовательно, *x* не будет больше изменяться.

Обычно нас не интересуют такие вот простые одномерные функции, поскольку в общем случае необходимо найти максимум многомерной функции. Для этого заменяем *x* на вектор $\vec{x}$ и наклон $f'(x)$ на градиент $\nabla f(\vec{x})$. В качестве напоминания: градиент – это просто вектор, в котором каждый элемент равен наклону $f'(x)$ в соответствующем измерении, т.е. $\left\langle \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right\rangle$. Таким образом, грубо говоря мы просто идем по наклону функции по всем измерениям одновременно. Вот алгоритм Градиентного Подъема по всей его пятистрочной сияющей славе:

---

[1] Перевод раздела из книги Luke S. Essentials of Metaheuristics. A Set of Undergraduate Lecture Notes. Zeroth Edition. Online Version 0.2. September, 2009 (http://cs.gmu.edu/~sean/book/metaheuristics/). Перевел – Юрий Цой, 2009 г.
Любые замечания, касающиеся перевода, просьба присылать по адресу yurytsoy@gmail.com
Данный текст доступен по адресу: http://qai.narod.ru/GA/meta-heuristics_1.pdf

[2] На самом деле обычно этот метод называют **градиентным спуском**, т.к. он используется для поиска *минимума* функции. Для этого необходимо *вычесть* градиент (наклон) вместо того, чтобы его *прибавлять*, т.е. в Алгоритме 1 строка изменится на $\vec{x} \leftarrow \vec{x} - a\nabla f(\vec{x})$. Однако, во всех последующих примерах будем всегда производить поиск максимума, поэтому будем последовательными и непротиворечивыми.

Algorithm 1 *Gradient Ascent*

1: $\vec{x} \leftarrow$ random initial vector
2: **repeat**
3: $\quad \vec{x} \leftarrow \vec{x} + \alpha \nabla f(\vec{x})$ ▷ In one dimension: $x \leftarrow x + \alpha f'(x)$
4: **until** $\vec{x}$ is the ideal solution or we have run out of time
5: **return** $\vec{x}$

Заметьте, что алгоритм работает пока мы не найдем «идеальное решение» или «закончится время». Откуда мы узнаем, что решение – идеальное. Обычно по наклону, который равен 0. Однако существуют функции, в которых помимо максимумов существуют точки, с точно таким же признаком: минимумы функции (конечно же) и седловые точки, подобные показанной на рис. 2.

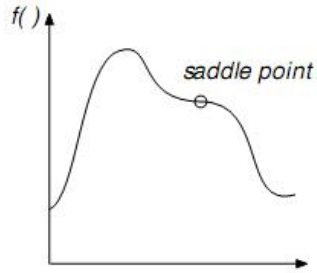

Рис. 2. Седловая точка

Одна из проблем градиентного подъема – это его время сходимости. Когда мы подберемся к максимуму функции, градиентный подъем просто перепрыгнет вершину и приземлится на другой стороне холма. Он может перепрыгивать вершину много раз, болтаясь туда-сюда в процессе приближения к максимуму. Эта ситуация показана на рис. 3.

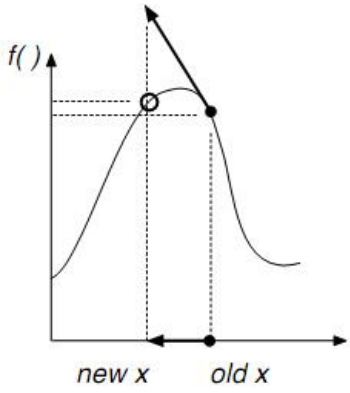

Рис. 3. Градиентный подъем, перепрыгивающий максимум

Одна из причин этого явления заключается в том, что размер прыжков градиентного подъема целиком зависит от текущего наклона. Если наклон очень крутой, то и прыжок будет большим, даже если это будет бессмысленно. Один из способов решения этой проблемы – настройка Градиентного Подъема для рассматриваемой задачи путем изменения значения a . Возьмем очень малое a , и тогда Градиентный Подъем не перепрыгнет максимум, хотя будет

очень долго взбираться по холму, пока не дойдет до вершины. При очень большом a Градиентный Подъем будет постоянно перепрыгивать вершину, что тоже приведет к росту времени сходимости к максимуму, если он вообще будет найден. Поэтому нам нужно выбрать значение a , про которое можно сказать «самое то».

Мы также могли бы модифицировать алгоритм, изменяя другие факторы. Например, если бы мы могли вычислять не только $f'(x)$, но $f''(x)$, то можно использовать **Метод Ньютона**[3]. Эта вариация на тему Градиентного Подъема включает дополнительно множитель $\frac{1}{f''(x)}$ вот так: $\vec{x} \leftarrow \vec{x} + a\frac{f'(\vec{x})}{f''(\vec{x})}$. Такая модификация будет уменьшать a по мере приближения к точке с нулевым наклоном. В многомерном случае ситуация уже сложнее. Многомерная версия первой производной $f'(x)$ это градиент $\nabla f(\vec{x})$. А вот многомерная версия *второй* производной $f''(x)$ представляет сложную матрицу, называемую **Гессианом** $H_f(\vec{x})$, которая состоит из частных вторых производных для каждого измерения:

$$H_f(\vec{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1}\frac{\partial f}{\partial x_1} & \frac{\partial}{\partial x_1}\frac{\partial f}{\partial x_2} & \mathbf{L} & \frac{\partial}{\partial x_1}\frac{\partial f}{\partial x_n} \\ \frac{\partial}{\partial x_2}\frac{\partial f}{\partial x_1} & \frac{\partial}{\partial x_2}\frac{\partial f}{\partial x_2} & \mathbf{L} & \frac{\partial}{\partial x_2}\frac{\partial f}{\partial x_n} \\ \mathbf{M} & \mathbf{M} & \mathbf{O} & \mathbf{M} \\ \frac{\partial}{\partial x_n}\frac{\partial f}{\partial x_1} & \frac{\partial}{\partial x_n}\frac{\partial f}{\partial x_2} & \mathbf{L} & \frac{\partial}{\partial x_n}\frac{\partial f}{\partial x_n} \end{bmatrix}.$$

Ситуация усложняется тем, что мы *делим* на вторую производную, что в многомерном случае означает поиск обратной матрицы. В целом метод выглядит следующим образом:

Algorithm 2 *Newton's Method (Adapted for Maximum Finding)*

1: $\vec{x} \leftarrow$ random initial vector
2: **repeat**
3: $\quad \vec{x} \leftarrow \vec{x} + \alpha[H_f(x)]^{-1}\nabla f(\vec{x})$ ▷ In one dimension: $x \leftarrow x + \alpha\frac{f'(x)}{f''(x)}$
4: **until** $\vec{x}$ is the ideal solution or we have run out of time
5: **return** $\vec{x}$

В силу использования второй производной Ньютоновский Метод обычно сходится быстрее, чем обыкновенный градиентный спуск, и кроме этого дает информацию, позволяющую определить, находимся ли мы в вершине локального максимума (чтобы не перепутать с минимумом или седловой точкой), т.к. в точке максимума $f'(x)$ равна нулю, а $f''(x)$ отрицательна.

Однако даже в этом случае мы не можем справиться с *по-настоящему серьезной* проблемой для подобных методов: они застревают в **локальных оптимумах**. Локальные оптимумы функции, это оптимумы (в нашем случае, максимумы) в локальной окрестности. Глобальны оптимум, это оптимум для всей функции. На рис. 4 показан путь Градиентного Подъема, который попался в локальный оптимум. Градиентный Подъем и Метод Ньютона являются **алгоритмами локальной оптимизации**.

---

[3] В честь Сэра Исаака Ньютона, 1642–1727. Этот метод обычно используется для поиска нулей (корней) функций, но его можно легко модифицировать для поиска максимума, как мы и сделали.

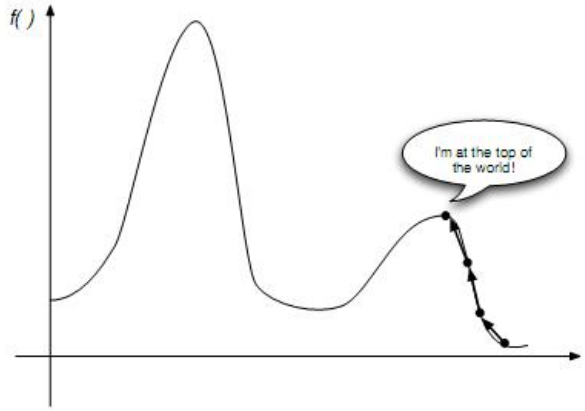

Рис. 4. Градиентный Подъем, застрявший в локальном оптимуме

Но как же избежать локального оптимума? Для методов, которые мы рассмотрели, существует только один выход: изменить значение a на достаточно большое, чтобы алгоритм потенциально смог не только перепрыгнуть вершину текущего холма, но и приземлиться на *другом холме*.

Либо мы могли бы запустить Метод Ньютона по большому циклу, снова и снова пытаясь найти решение при различных начальных точках, и возвратив лучший найденный результат. Чтобы определить «лучший найденный результат» необходимо уметь вычислять значение $f(x)$ (до сих пор это не было необходимо), чтобы можно было сравнивать различные результаты. Предполагая, что это возможно, можем теперь получить **алгоритм глобальной оптимизации**:

Algorithm 3 *Newton's Method with Restarts (One Dimensional Version)*

$x \leftarrow$ random initial value
$x^* \leftarrow x$ ▷ x^* will hold our best discovery so far.
repeat
  repeat
    $x \leftarrow x + \alpha \frac{f'(x)}{f''(x)}$
  until $f'(x) = 0$ and $f''(x) > 0$ ▷ That is, until we're at the maximum.
  if $f(x) > f(x^*)$ **then** ▷ Found a new best result!
    $x^* \leftarrow x$
  $x \leftarrow$ random value
until we have run out of time
return x^*

Алгоритм глобальной оптимизации гарантирует нахождение глобального оптимума, если будет работать в течение достаточно длительного времени. Однако вышеприведенный алгоритм по-настоящему глобален только в теории: значение $f'(x)$ никогда не будет в точности равным 0. Поэтому нам нужно состряпать этот случай: если $-e < f'(x) < e$ для некоторого очень малого e, то будем считать, что $f'(x)$ находится «достаточно близко к нулю».

В качестве упражнения предоставим читателю придумать многомерную версию этого алгоритма.