Оптимизация негладких функций

1. Оптимизация негладких функций

Оптимизационные задачи встречаются очень часто в различных областях. Пусть используется алгоритм машинного обучения с параметрами $\alpha_1, \ldots, \alpha_N$. Параметрами являются некоторые числа, которые необходимо подобрать таким образом, чтобы качество работы алгоритма было оптимальным.

Качество решения задачи можно рассмотреть как функцию параметров алгоритма. Таким образом, нужно найти решение следующей оптимизационной задачи:

$$Q\big(\alpha_1, \ldots, \alpha_N\big) \to \max_{\alpha_1,\ldots,\alpha_N}.$$

Одним из методов численной оптимизации является *градиентный спуск*. Однако, функция от параметров может не иметь градиента (*негладкая функция*). К тому же, градиент бывает достаточно сложно вычислить. Существует также проблема *локальных минимумов*. Для получения наименьшего значения функции необходимо достичь глобального минимума, что в случае градиентного спуска не всегда просто — градиент и в точке локального минимума равен нулю (см. рис. 1).

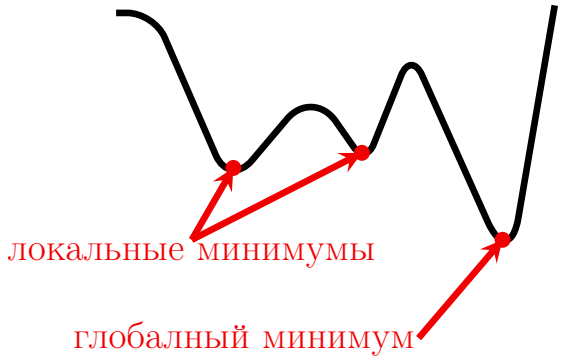

Рис. 1.

Более подходящим методом является *случайный поиск*. При минимизации функции на каждом шаге выбираются следующие значения параметров случайно, но при этом «хорошие» значения параметров выбираются с большей вероятностью, и тогда в среднем минимум будет приближаться.

Например, если нет возможности вычислить значение градиента (или если функция негладкая), то можно попробовать следующий метод: на нулевом шаге фиксируется некоторый небольшой параметр d. На первом шаге выбирается вектор $\vec{u}$ из равномерного распределения на единичной сфере. На втором шаге вычисляется значение $f\big(x + d \cdot \vec{u}\big)$ и производится сдвиг в направлении вектора u на величину:

$$\frac{f(x) - f\big(x + d \cdot \vec{u}\big)}{d}.$$

Можно заметить, что в приведенном методе ни на каком шаге не вычисляется градиент, тем не менее в среднем движение происходит в направлении антиградиента или, в случае его отсутствия — в направлении антиградиента сглаженной функции.

Параметр d влияет на то, как сильно сглаживается функция. К примеру, на рис. 2 изображена исходная функция, а на рис. 3 видно, как увеличивается сглаживание с увеличением d.

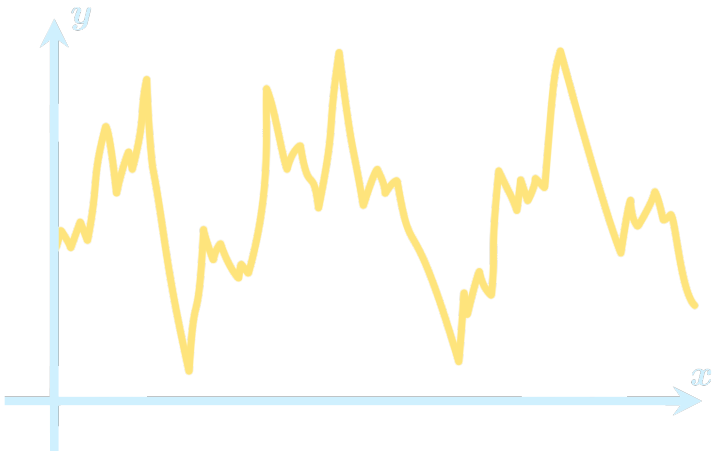

Рис. 2.

Рис. 3.

2. Метод имитации отжига

Среди методов глобальной оптимизации имеется *метод имитации отжига* (*алгоритм Метрополиса*) — один из вариантов случайного поиска. Данный метод не нуждается в гладкости функции.

Метод был вдохновлен физическим процессом, происходящим при отжиге металлов — переходы атомов из одной ячейки кристаллической решетки в другую происходят, если в результате перехода энергия атомов уменьшится, причем с падением температуры вероятность переходов падает. Этот процесс является, по сути, природной оптимизацией.

Алгоритм заключается в следующем. Начинают в любой случайной точке и задают начальное значение температуры $T = T_0$. Следующая точка выбирается недалеко от предыдущей случайным образом. Если значение в этой точке меньше, чем в предыдущей (для задачи минимизации), то переходят в неё (см. рис. 4).

Если же значение в новой точке больше, чем в предыдущей, то переход в нее обеспечивает возможность выбираться из локальных минимумов. Переход в эту точку будет осуществляться с вероятностью (см. рис. 5):

$$P = \exp\left(-\frac{f(x^* - f(x_0))}{T}\right).$$

Необходимо отметить, что с ростом температуры эта вероятность становится все меньше.

После завершения первого шага значение температуры обновляется ($T = T_1$), причем T_k должны быть положительными и убывать с ростом k. Шаги необходимо повторять до тех пор, пока температура снижается (или решение улучшается).

При решении задач оптимизации не всегда необходимо найти минимум — бывает, что нужно найти значение меньше, чем известное. Например, в задаче оптимизации

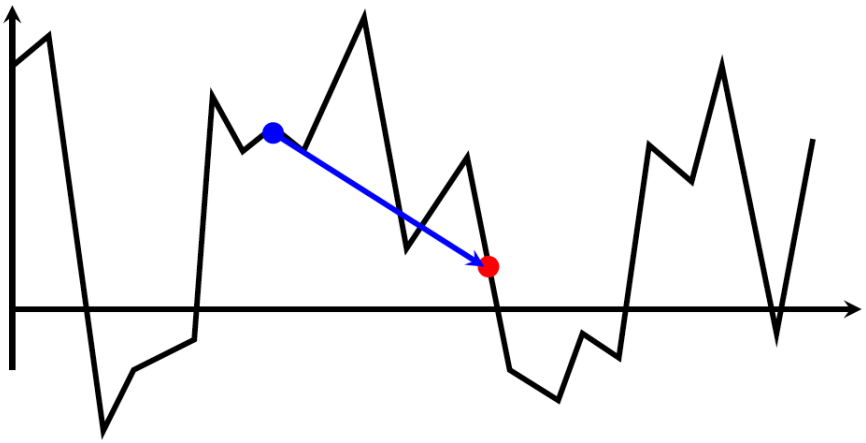

Рис. 4.

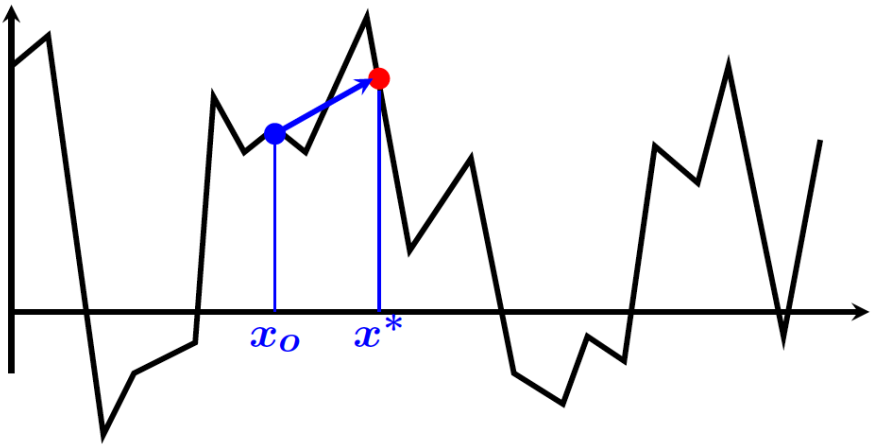

Рис. 5.

рекламной кампании обычно достаточно уменьшить затраты, и при этом необязательно доказывать, что меньше затратить невозможно. В рассматриваемом алгоритме при некоторых способах выбора следующей точки можно гарантировать, что алгоритм будет находить, по крайней мере, точку лучше начального приближения.

Метод имитации отжига помогает при оптимизации достаточно сложных функций, для которых неприменимы градиентные методы. Реализацию данного метода можно взять в библиотеке *SciPy* в модуле *optimize* языка *Python* (функция *anneal*).

3. Генетические алгоритмы

Генетические алгоритмы — это методы оптимизации, вдохновленные процессом эволюции. Они моделируют данный процесс, поэтому содержат в себе стадии генерации популяции и повторяемые стадии мутации, скрещивания и отбора. Порядок шагов может варьироваться.

Дифференциальная эволюция — это алгоритм, который используется для оптимизации функций от вещественных переменных. Вначале происходит генерация n случайных векторов, называемых *популяцией* (см. рис. 6).

Затем выбирается некоторый вектор из популяции и еще три случайных вектора (см. рис. 7).

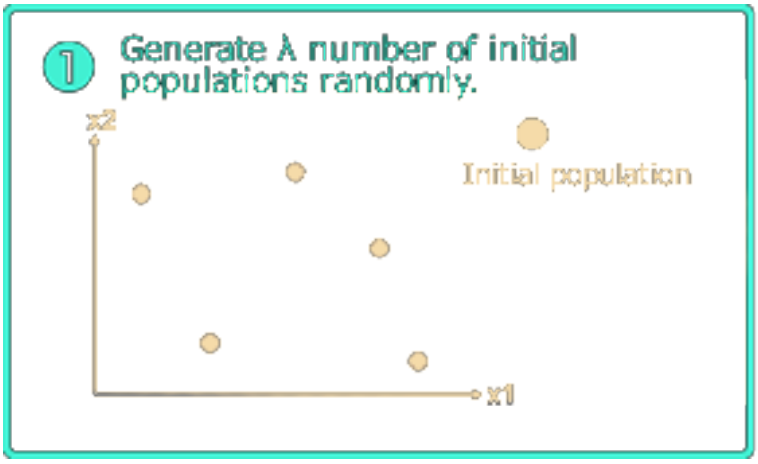

Рис. 6.

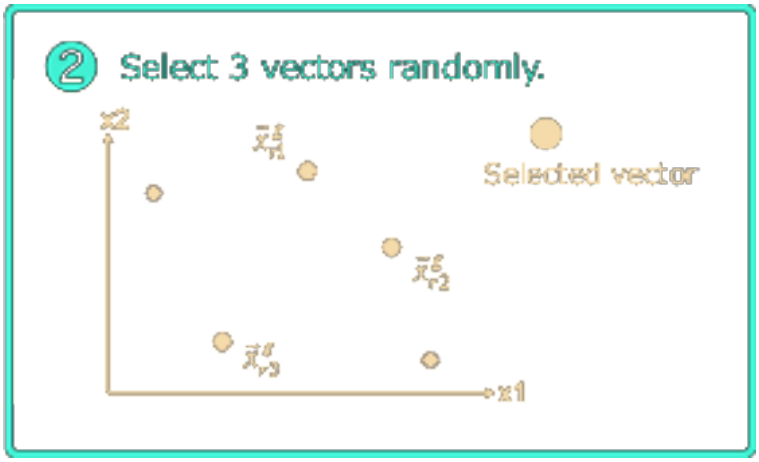

Рис. 7.

Вычисляется разность между двумя выбранными случайно векторами. В направлении этой разности производится сдвиг от третьего вектора (см. рис. 8). Итоговый вектор называется *мутантным*, а данная операция называется *мутацией*.

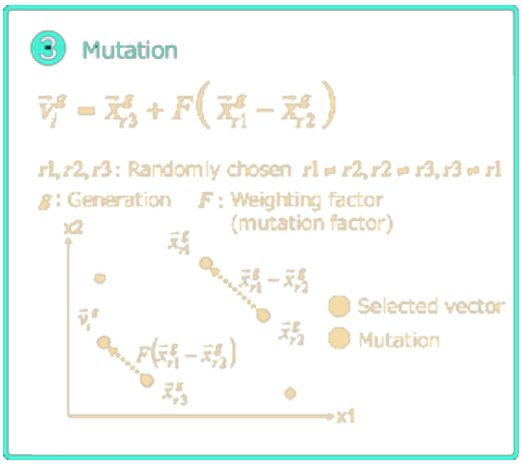

Рис. 8.

Далее производится операция *скрещивания*: с вероятностью p выбирается коор-

дината от первого родителя, с вероятностью $1 - p$ — от второго (и так для всех координат). Таким образом, потомок обладает свойствами обоих родителей (см. рис. 9).

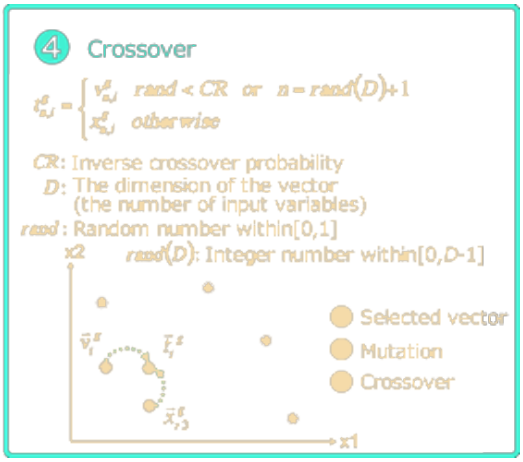

Рис. 9.

Следующая стадия — *отбор*. Существует много вариантов проведения этой стадии. К примеру, если сын оказался лучше родителя, то в популяции оставляют его, в противном случае оставляют родителя. Другой вариант — сначала проделываются мутации и скрещивания для всей популяции, затем отбирают N лучших.

После прохода по всей популяции генерируется новая, для которой повторяются те же стадии. Данные действия повторяются до сходимости. Этот метод оптимизации находит значения функции, в которых она будет меньше, чем в начальных приближениях.

В случае оптимизации функции от векторов, состоящих из 0 и 1, операция мутации из дифференциальной эволюции может привести к появлению вектора, состоящего не из 0 и 1. Можно изменить данную операцию следующим образом: вместо смещения на разность каждую координату заменяют с вероятностью R (*параметром алгоритма*). Операция скрещивания остается без изменений.

Для лучшей работы методов, основанных на эволюционных идеях, иногда применяются некоторые «трюки». Например, можно создавать несколько «островков» из разных популяций и применять алгоритм для каждой из них. Здесь нужно учитывать, что с ростом числа координат метод резко начинает работать медленней.

Нужно также иметь в виду, что могут быть разные операторы скрещивания и мутации. На данную тему выполнялось множество экспериментов и существует много статей.

Генетические алгоритмы можно использовать для задачи глобальной оптимизации. Также необходимо заметить, что они не требовательны к гладкости функции.

Реализация в *Python* — пакет *SciPy*, модуль *optimize*, метод *differential evolution*.

4. Метод Нелдера – Мида

Метод Нелдера – Мида применяется для оптимизации функций, в том числе негладких и зашумленных. Данный метод устроен очень просто, и его легко реализовать. Этот метод используется по умолчанию в функции *minimize* из *scipy.optimize*.

Начало работы алгоритма выглядит следующим образом. В случае оптимизации функции от n переменных выбираются $n+1$ начальных точек. Желательно выбирать точки в той области, про которую уже известно, что значения функции в ней не слишком далеки от предполагаемого экстремума.

Данные точки образуют *симплекс*. Для прямой симплексом является отрезок, для плоскости — треугольник, для трехмерного пространства — тетраэдр. В n-мерном случае симплекс — это n-мерное обобщение тетраэдра.

В двумерном случае линии уровня функции и симплекс связаны следующим образом. Рассматривается треугольник, построенный на трех выбранных точках. Далее этот треугольник деформируют таким образом, чтобы он постепенно приближался к минимуму и стягивался вокруг него. Именно из-за такого поведения этот метод иногда называют *амебой*.

Таким образом, дальнейшие шаги в процессе деформации — это отражение, сжатие, растяжение, сжатие симплекса. Данный метод особенно хорошо работает, если использовать его после грубого метода (например, поиска по сетке) для уточнения экстремума.