Теория оптимизации

Оптимизация негладких функций. Метод имитации отжига. Генетические алгоритмы. Алгоритм дифференциальной эволюции. Алгоритм Нелдера-Мида.

Даниил Корбут
Специалист по Анализу Данных

НЕТОЛОГИЯ

Часть 2: Лекция 3

Даниил Корбут
DL Researcher
Insilico Medicine, Inc

Окончил бакалавриат ФИВТ МФТИ (Анализ данных) в 2018г
Учусь на 2-м курсе магистратуры ФИВТ МФТИ
Работал в Statsbot и Яндекс. Алиса.
Сейчас в Insilico Medicine, Inc, занимаюсь генерацией активных молекул и исследованиями старения с помощью DL.

Градиентный спуск (из прошлой лекции)

Это итерационный метод. Решение задачи начинается с выбора начального приближения $\vec{x}^{[0]}$

После вычисляется приблизительное значение $\vec{x}^{1}$ затем $\vec{x}^{2}$ и так далее

!

$$\vec{x}^{[j+1]} = \vec{x}^{[j]} - \gamma^{[j]} \nabla F(\vec{x}^{[j]}), \qquad \text{где } \gamma^{[j]} \text{ — шаг градиентного спуска.}$$

Идея: идти в направлении наискорейшего спуска, а это направление задаётся антиградиентом $-\nabla F$

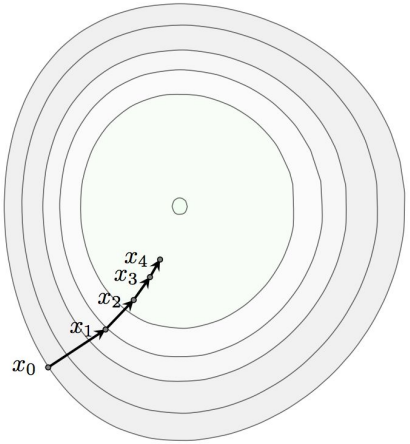


Рис. 3: Градиентный спуск

https://www.coursera.org/learn/mathematics-and-python

Оптимизация негладких функций

Иногда даже если градиент у интересующей функции существует, часто оказывается, что вычислять его непрактично.

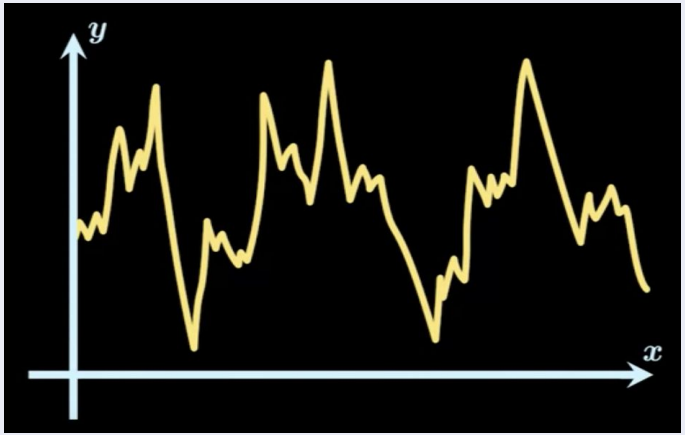
Пусть есть алгоритм с параметрами $\alpha_1, ..., \alpha_N$

Задача: подобрать параметры так, чтобы алгоритм давал лучший результат.

Можно записать в виде оптимизационной задачи

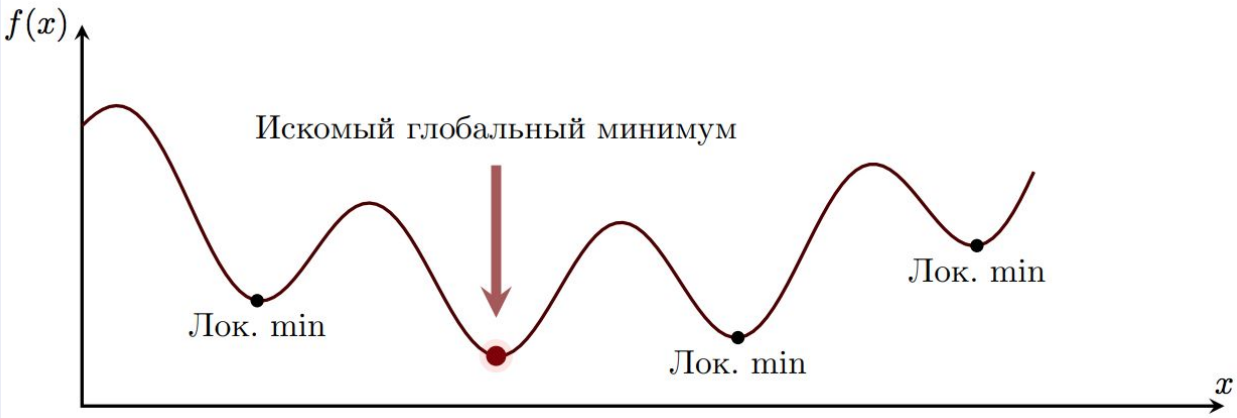
$$Q(\alpha_1, ..., \alpha_N) \to \max_{\alpha_1, ..., \alpha_N}$$

Вычисление градиента в этом случае зачастую невозможно в принципе или крайне непрактично.



Градиентный спуск без градиента

Другая проблема градиентных методов — проблема локальных минимумов.



Градиентный спуск, попав на дно локального минимума, где градиент также равен нулю, там и остается. Глобальный минимум так и остается не найденным.
Решить эти проблемы позволяют **методы случайного поиска**.
Общая идея этих методов заключается в намеренном введении элемента случайности.

Градиентный спуск без градиента

Дано:

$f(\vec{x}) \to \min_{\vec{x}}$

d — параметр метода.

Давайте решать задачу оптимизации для функции, у которой нельзя явно вычислить градиент.

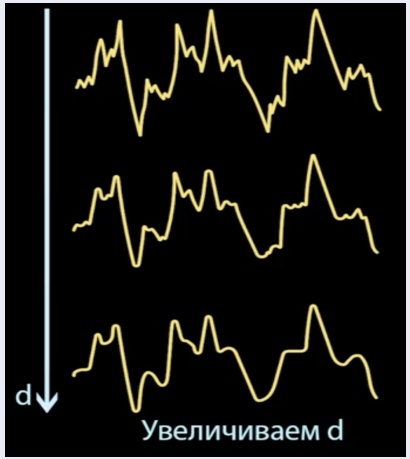
Сначала выбирается $\vec{u}$ (случайный вектор $\vec{u}$ равномерно распределен по сфере).

Затем вычисляем численную оценку производной по направлению $\dfrac{f(\vec{x}) - f(\vec{x} + d\vec{u})}{d}$

!

Сдвигаем точку в направлении $\vec{u}$ пропорционально величине с прошлого шага.

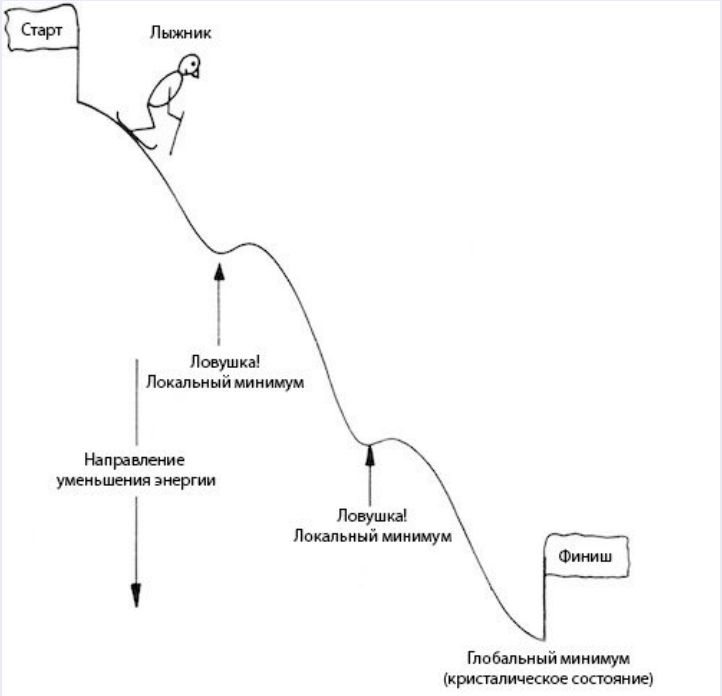
Следует отметить, что ни на каком шаге градиент функции не вычисляется.
Величина смещения зависит от выражения функции в точке $\vec{x} + d\vec{u}$.
и в среднем смещение происходит смещение по антиградиенту сглаженной функции.
Число d - «параметр сглаживания» при нахождении численной оценки производной.



Метод имитации отжига

Алгоритм оптимизации, для которого **не требуется гладкость функции**.
Алгоритм основывается на имитации физического процесса, который происходит при кристаллизации вещества.
Цель отжига — привести систему, которой является образец металла, в состояние с минимальной энергией.

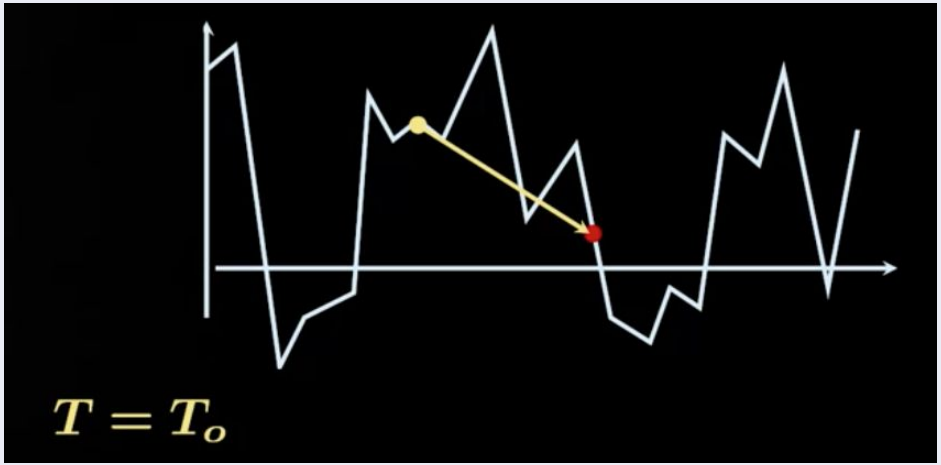
Для задач оптимизации имитация процесса может быть произведена следующим образом.
Вводится параметр T, который имеет смысл температуры, и в начальный момент ему устанавливается значение T0.
Набор переменных, по которым происходит оптимизация, будет обозначаться как x.

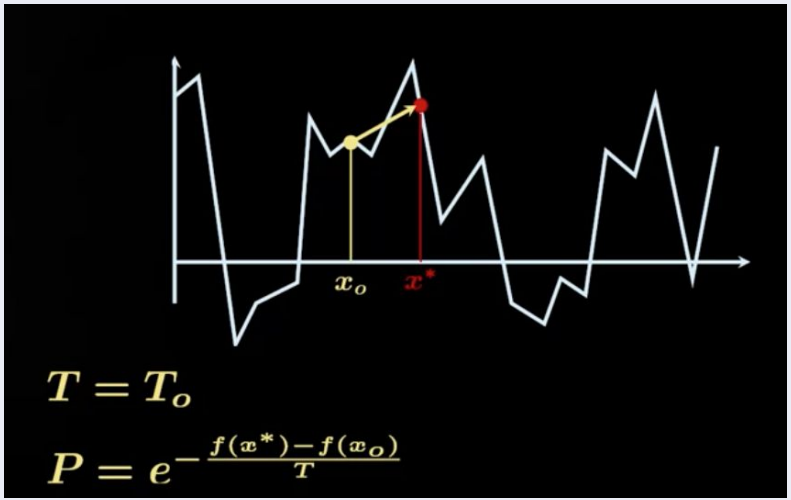


https://www.pvsm.ru/matematika/53102

Метод имитации отжига

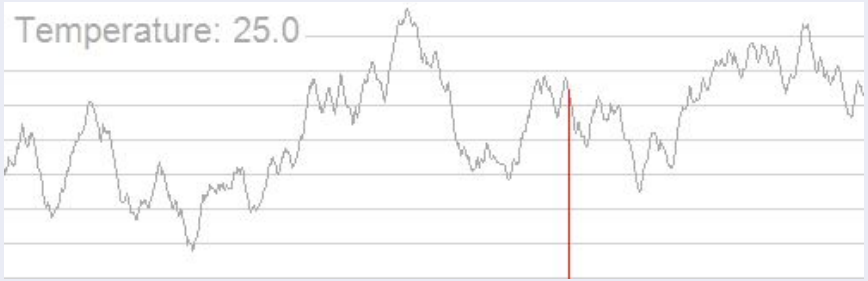
1. Начальное приближение - произвольная точка.
2. Выбираем x^* случайно из множества соседних состояний.
3. Если значение функции меньше, то переходим в x^*
4. Если значение функции больше, то переходим в x^* с вероятностью $P = e^{-\frac{f(x^*)-f(x)}{T}}$
5. Уменьшаем значение температуры



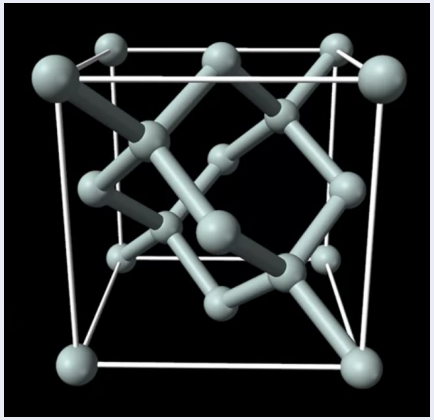


Метод имитации отжига

Благодаря переходам в худшее состояние в методе имитации отжига удалось решить проблему локальных минимумов и не застревать в них. Постепенно, при уменьшении температуры, уменьшается и вероятность переходов в состояния с большим значением функции. Таким образом в конце имитации отжига в качестве x оказывается искомый глобальный минимум.



https://en.wikipedia.org/wiki/Simulated_annealing



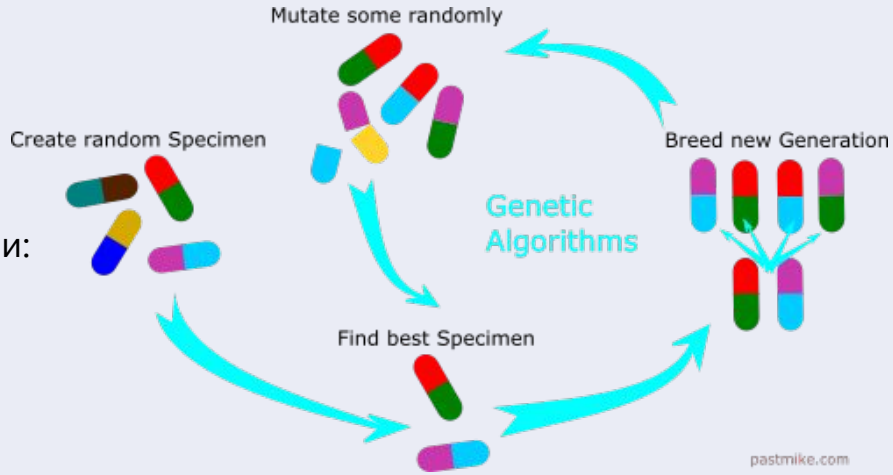
Генетические алгоритмы

Генетические алгоритмы моделируют процесс естественного отбора в ходе эволюции и являются еще одним семейством методов оптимизации.

Генетические алгоритмы включают в себя стадии:

1. генерации популяции
2. мутаций
3. скрещивания
4. отбора

Порядок стадий зависит от конкретного алгоритма.



Алгоритм дифференциальной эволюции

Для оптимизации функции $f(\vec{x})$ вещественных переменных применяется алгоритм дифференциальной эволюции.

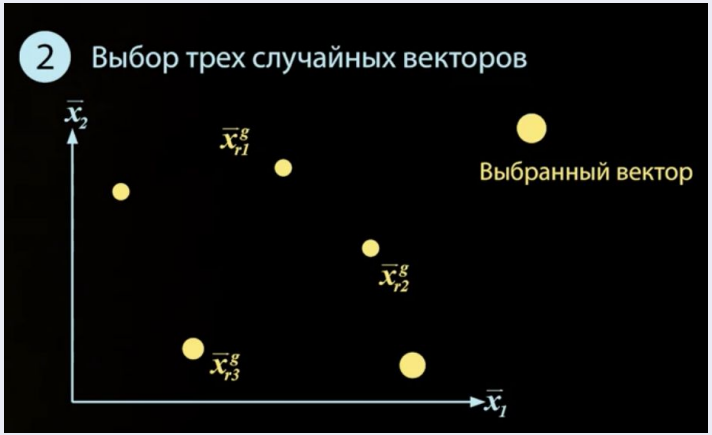
Популяция - множество векторов $\mathbb{R}^n$

Размер популяции - N

Сила мутации - F [0, 2]

Вероятность мутации - P

В качестве начальной популяции выбирается набор из N случайных векторов. На каждой следующей итерации алгоритм генерирует новое поколение векторов, комбинируя векторы предыдущего поколения.

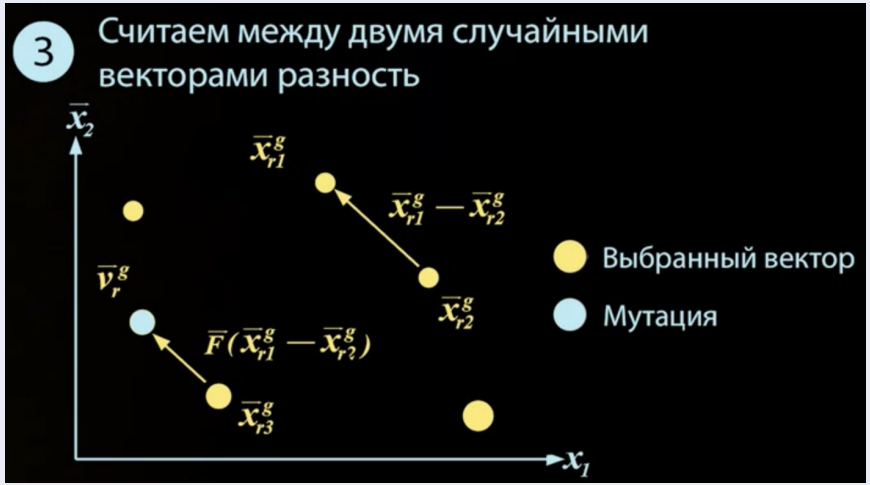


Алгоритм дифференциальной эволюции

На каждой итерации для каждого вектора x_i выбираются 3 неравные ему вектора v_1, v_2, v_3.
На основе этих векторов генерируется так называемый **мутантный** вектор:

$$v = v_1 + F \cdot (v_2 - v_3)$$

На **стадии скрещивания** каждая координата мутантного вектора с вероятностью p замещается соответствующей координатой вектора xi . Получившийся вектор называется пробным.



Алгоритм дифференциальной эволюции

На **стадии отбора**, если пробный вектор оказывается лучше исходного xi (то есть значение исследуемой функции на пробном векторе меньше, чем на исходном), то в новом поколении он занимает его место.

Если сходимость не была достигнута, начинается новая итерация.

Рассмотренный алгоритм хорошо применим для функций, зависящих от векторов действительных чисел. Часто встречаются зависимости от бинарных векторов.

- Вариант 1. Если сын лучше родителя — оставляем его, если нет — родителя.
- Вариант 2. Сначала проделать мутации и скрещивания для всей популяции, а потом отобрать N лучших.

Алгоритм дифференциальной эволюции

- В случае бинарных векторов можно определить **мутацию** следующим образом: с вероятностью **p** менять 0->1 и 1->0 в исходных векторах. **Скрещивание** без изменений

- Часто, чтобы увеличить эффективность алгоритма, создаются несколько независимых популяций. Для каждой такой популяции формируется свой начальный набор случайных векторов. В таком случае появляется возможность использовать генетические алгоритмы для решения задач глобальной оптимизации.

- Эффективность метода зависит от выбора операторов мутации и скрещивания для каждого конкретного типа задач.

Метод Нелдера-Мида

Метод Нелдера-Мида, или метод деформируемого многогранника, применяется для нахождения решения задачи оптимизации вещественных функций многих переменных. На каждой итерации для функции от **n** переменных требуется вычислить значение в **n+1** точке.

Метод прост в реализации и полезен на практике, но, с другой стороны, для него не существует теории сходимости — алгоритм может расходиться даже на гладких функциях.

Используется по умолчанию в функции **minimize** из **scipy.optimize**.

Метод Нелдера-Мида

1) выбираем n+1 точку, образующие симплекс

Рис. 2: Отрезок (1-симплекс)

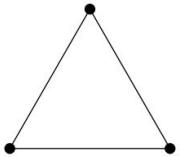


Рис. 3: Треугольник (2-симплекс)

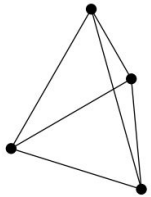
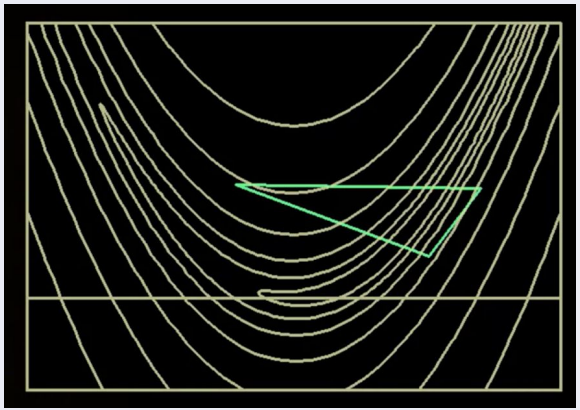


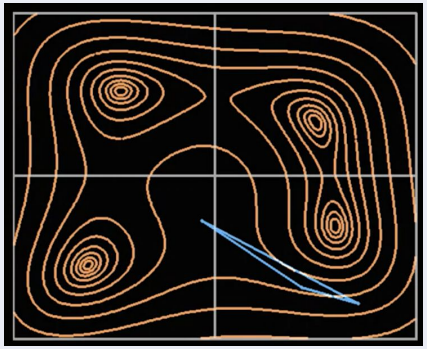
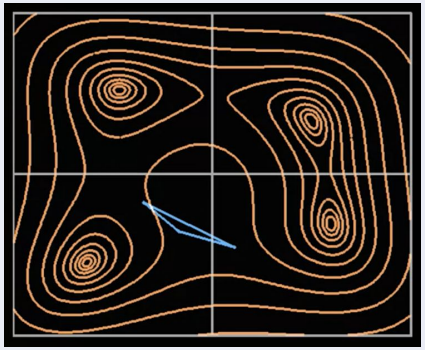
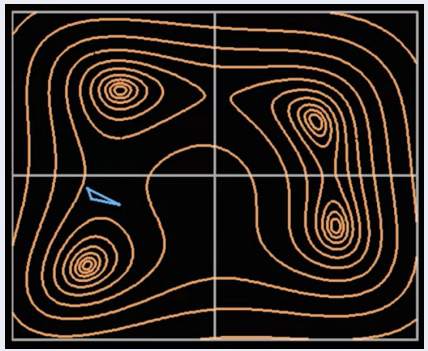
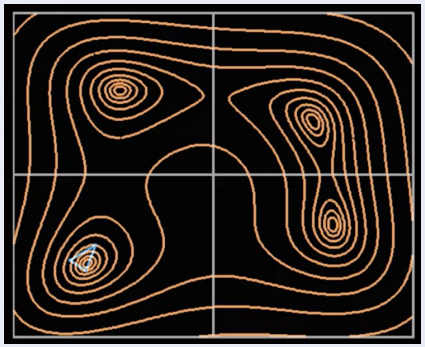
Рис. 4: Тетраэдр (3-симплекс)

2) сравниваем значения функции в вершинах



Метод Нелдера-Мида

3) деформируем (отражение, сжатие, растяжение, сжатие симплекса) треугольник так, чтобы он “подползал” к минимуму функции

Спасибо за внимание!