Magzyan1

mrGCrusader, micromolecul, Matveiiy

January 2025

1 Градиентный спуск

Градиентный спуск - важный алгоритм, примениемый в множестве сфер computer-science. Этот метод приминяется для дифференцируемых, унимодальных функций, действующих в пространство $\mathbb{R}$. Он работает следующим образом:
1) Выбираем точку, для старта нашего алгоритма
2) Вычисляем градиент функции в этой точке(мы можем так сделать, так как функция дифференцируема)
3) Идём по направлению антигриента на величину равную $\|h(m) * \nabla f\|$, где h - величина, на которую мы переходим на m шаге(learning rate scheduling), он может зависеть от нескольких параметров.
4) Проверяем выполнение критерия остонова, он может быть разным, и зависеть от нескольких параметров. Если критерий выполнен, то мы останавливаем наш алгоритм(вероятно мы нашли приближённую к минимуму функцию), если не выполнен, то переходим к шагу 2.

2 Критерии останова

Наша команда реализовала несколько критериев останова от самых простых, до самых сложных, вот некоторые из них:
1) MaxIterations - критерий, выполняющийся, когда мы проделали шаг 2 n раз. Этот критерий довольно прост в реализации, однако при использовании этого критерия есть риск перевычисления.
2) Convergence - критерий, выполняющийся, когда разница в значениях функции на n и $n-1$ шаге меньше заданного нами параметра ϵ. Этот критерий также прост в реализации, однако алгоритм может не выполнятся бесконечное количество времени(например если передали плохую функцию без минимума).
3) GradientNormComparative - критерий, выполняющийся, когда норма градиента меньше заданного нами значения $\epsilon * \|\nabla f(x_0)\|$ Этот критерий также может быть бесконечно неверен, если нам передали функцию без минимального значения.
Также мы реализовали класс Combine, способный комбинировать критерии останова. Комбинация критериев - важная механика, помогающая регулировать безопастность и эффективность алгоритма.

3 Шаг спуска

Также важным этапом реализации являлся выбор learning rate scheduling. Вот некоторые варианты из тех, которые мы реализовали:
1) Constant - шаг спуска равен заданной нами константе, этот шаг может привести к некоторым ошибкам, связанным с точностью, обычно используют монотонно убывающие функции, для

большей точности вычислений, однако это также повышает и объём вычислений, для нахождения ответа.
Далее представленны методы, которые просто являются монотонно убывающими функциями от шага k:
2) TimeBasedDecay - шаг спуска равен: $h(k) = \frac{h_0}{(1+\lambda*k)}$ - шаг с ассимтотикой $o(k^{-1})$
3) ExponentialDecay - шаг спуска равен: $h(k) = h_0 * exp(-\lambda * k)$ - шаг с ассимтотикой $o(exp(-\lambda * k))$

4 Наискорейший градиентый спуск

Наискорейший градиентный спуск - обычный градиентный спуск с особым шагом спуска, который предполагает движение до тех пор, пока мы не достигнем минимума функции f на этом направлении. Мы реализовали это с помощью различных реализаций одномерного спуска: Фибоначи, Армихо, Гольштейна, используя scipy.optimize.

5 Функции

Теперь перейдём к примерам функций, иллюстрирующих работу спуска: