Лабораторная работа 2: Использование численных методов в задачах оптимизации

Цель работы: Знакомство с современными методами оптимизации, применение численных методов дифференцирования для решения прикладных задач.

Описание: В рамках лабораторной работы предлагается использовать численные методы, включающие в себя конечно-разностные схемы вычисления производных, для задач оптимизации (без ограничений).

Предлагаемые методы: краткая справка по алгоритму BFGS

В качестве первого алгоритма предлагается использовать модифицированный метод BFGS. Сходимость оптимизационного алгоритма гарантируется только на дважды-дифференцируемой выпуклой функции.

Общая постановка оптимизационной задачи (минимизации): пусть $\mathbf{x} = (x_1, \ ... \ x_n) \in \mathbb{R}$, а допустимое множество составляет всё $\mathbb{R}$, т.е. ограничения не ставятся. Задана целевая функция $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$, для которой необходимо выполнить поиск: $f(\mathbf{x}) \to min$. Иначе задачу оптимизации можно представить так: требуется найти решения $\mathbf{x}' : f(\mathbf{x}') \leq f(\mathbf{x}^*) \ \forall x^* \in \mathbb{R}^n$.

В качестве алгоритма оптимизации предлагается использовать классический алгоритм Бройдена — Флетчера — Гольдфарба — Шанно (BFGS), где используется следующая модель целевой функции на k-ой итерации:

$$f(\mathbf{x}_k + \mathbf{p}) = f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)\mathbf{p} + \frac{1}{2}\mathbf{p}^T H(\mathbf{x}_k)\mathbf{p} \tag{1}$$

В данном соотношении $\mathbf{p}$ - минимизатор, направление, в котором движется оптимизационный алгоритм; $H(\mathbf{x}_k)$ - матрица Гессе функции f. Для вычисления направления p_k, направления p на k-ой итерации используется следующая модель:

$$\mathbf{p}_k = -H_k^{-1}\nabla f_k \tag{2}$$

Для вычисления следующего приближаемого решения $\mathbf{x}_{k+1}$ используется соотношение $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$, где α - параметр, удовлетворяющий условиям Вольфе из ур. 3:

$$\begin{gathered} f(\mathbf{x}_k + \alpha_k \mathbf{p}_k) \leq f(\mathbf{x}_k) + c_1 \alpha_k \nabla f(\mathbf{x}_k)^T \mathbf{p}_k \\ \nabla f(\mathbf{x}_k + \alpha_k \mathbf{p}_k)^T \mathbf{p}_k \geq {}_2\alpha_k \nabla f(\mathbf{x}_k)^T \mathbf{p}_k \end{gathered} \tag{3}$$

Использование алгоритма линейного поиска

Параметр α_k соответствует величине шага вдоль направления поиска и определяется так, чтобы соблюдать условия Вольфе. Для произвольных параметров $c_1, \ c_2$, $0 < c_1 < c_2 < 1$, а также заданного $\mathbf{p}$-направления поиска процедура подбора происходит по схеме на алг. 1:

Для определения значений α в интервале от α_i до α_j используется алгоритм приближений $\mathbf{zoom}(\alpha_i, \ \alpha_j)$, рассмотренный на алгоритме 2.

Детально процесс подбора параметров шага минимизатора описан в 3 главе [2].

Задание матрицы Гессе

В прикладных решениях задачи используется лишь приближенные значения H, так как вычислять точное значение определителя матрицы Гессе требует слишком

input : Оптимизируемая функция f, её градиент ∇f, приближение решения $\mathbf{x}$, направление минимизатора p, параметры $c_1,\ c_2,\ \alpha_{max},\ max\_iter$
output: Значение параметра a
$\alpha_0 \longleftarrow 0$, выбрать $\alpha_1 \in (0, \alpha_{max})$;
$i \longleftarrow 1$;
while *True* **do**
  Вычислить $f(\mathbf{x} + \alpha_i \mathbf{p})$;
  if $f(\mathbf{x} + \alpha_i \mathbf{p}) > f(\mathbf{x}) + c_1 \alpha_i \nabla f(\mathbf{x})$, *или* $(f(\mathbf{x} + \alpha_i \mathbf{p}) \geq f(\mathbf{x} + \alpha_{i-1}\mathbf{p})$ *и* $i > 1)$ **then**
    return zoom(α_{i-1}, α_i);
  Вычислить $\nabla f(\mathbf{x} + \alpha_i \mathbf{p})$;
  if $|\nabla f(\mathbf{x} + \alpha_i \mathbf{p})| \leq -c_2 |\nabla f(\mathbf{x})|$ **then**
    return α_i ;
  if $\nabla f(\mathbf{x} + \alpha_i \mathbf{p}) \geq 0$ **then**
    return zoom(α_{i-1}, α_i);
  Выбрать $\alpha_{i+1} \in (\alpha_{i+1}, \alpha_{max})$;
  $i \longleftarrow i + 1$;
end

Algorithm 1: Согласованный со слабым условием Вольфе алгоритм линейного поиска величины шага.

input : Оптимизируемая функция f, её градиент ∇f, приближение решения $\mathbf{x}$, направление минимизатора p, величины шага α_{low}, α_{high}, параметры c_1, и c_2
output: Значение параметра a
while *True* **do**
  Интерполировать f для определения α_j между α_{low} и α_{high}. Упрощенный подход: $\alpha_j \longleftarrow 0.5(\alpha_{low} + \alpha_{high})$;
  Вычислить $f(\mathbf{x} + \alpha_j \mathbf{p})$;
  if $f(\mathbf{x} + \alpha_j \mathbf{p}) > f(\mathbf{x}) + c_1 \alpha_j \nabla f(\mathbf{x})$ *или* $f(\mathbf{x} + \alpha_j \mathbf{p}) \geq f(\mathbf{x} + \alpha_{low}\mathbf{p})$ **then**
    $\alpha_{high} \longleftarrow \alpha_j$;
  else
    Вычислить $\nabla f(\mathbf{x} + \alpha_j \mathbf{p})$;
    if $|\nabla f(\mathbf{x} + \alpha_j \mathbf{p})| \leq -c_2 \nabla f(\mathbf{x}) \cdot \mathbf{p}$ **then**
      return α_j ;
    if $\nabla f(\mathbf{x} + \alpha_j \mathbf{p})(\alpha_{high} - \alpha_{low}) \geq 0$ **then**
      $\alpha_{high} \longleftarrow \alpha_{low}$;
    $\alpha_{low} \longleftarrow \alpha_j$;
  end
end

Algorithm 2: Процедура **zoom** для определения величины α_j.

много вычислений. Также, последующие вычисления требуют обратную матрицу Гессе, поэтому приближать рекомендуется именно её. На первом шаге эволюционного алгоритма матрицу допустимо задать как единичную, в то время как на последующих шагах для обновления значений используется соотношение 5, где $\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$ и $\mathbf{y}_k = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$:

$$H_{k+1} = (I - \rho_k \mathbf{s}_k \mathbf{y}_k^T) H_k (I - \rho_k \mathbf{s}_k \mathbf{y}_k^T) + \rho_k \mathbf{s}_k \mathbf{s}_k^T \tag{4}$$

$$\rho_k = \frac{1}{\mathbf{y}_k^T \mathbf{s}_k} \tag{5}$$

Критерием останова может выступать условие достижение достаточно низкого значения нормы градиента функции: $||\nabla f(\mathbf{x}_k)||_2 < \epsilon$.

Вариант алгоритма L-BFGS, позволяющий экономить вычислительные ресурсы за счёт иного подхода к приближению обратной матрицы Гессе. О данном методе можно прочесть в источниках [2], или [4].

Метод конечных разностей

Для градиента функции $\nabla f(\mathbf{x}_k) = (\frac{\partial}{\partial x_1} f(\mathbf{x}),\ \ldots\ , \frac{\partial}{\partial x_n} f(\mathbf{x}))$ можно использовать следующую конечно-разностную форму (6) для схемы вперёд, имеющей 1ый порядок аппроксимации, и (7) для центральной схемы, имеющей 2ой порядок аппроксимации:

$$\frac{\partial}{\partial x_j} f(\mathbf{x}) \approx \frac{f(\mathbf{x} + h_j \mathbf{e}_j) - f(\mathbf{x})}{h_j} \tag{6}$$

$$\frac{\partial}{\partial x_j} f(\mathbf{x}) \approx \frac{f(\mathbf{x} + h_j \mathbf{e}_j) - f(\mathbf{x} - h_j \mathbf{e}_j)}{2h_j} \tag{7}$$

Ход работы:

1. Для иллюстрации возможностей подхода вам предстоит искать минимум функции (8), где $\mathbf{x} \in \mathbb{R}^3$.

$$f(\mathbf{x}) = \frac{1}{2}[(x_1)^2 + \sum_{i=1}^{2}(x_i - x_{i+1})^2 + (x_3)^2] - x_1 \tag{8}$$

   - Покажите, что алгоритмы BFGS и L-BFGS можно применять к этой оптимизационной задаче.
   - Реализовать алгоритмы оптимизации BFGS и L-BFGS, в которых для каждой воплощены версии, в которых а) используется аналитическое (определите самостоятельно) задание градиента целевой функции, и б) численное, на основе конечных разностей. Не допускается использование готовых решения (например, можно использовать numpy при написании кода на python, однако не допускается использование scipy.optimize).
   - Реализовать инициализацию приближения матрицы Гессе через произвольную матрицу (например, единичную), и через вычисленные на основе конечных разностей частные производные целевой функции.
   - Сравнить поведение созданных вариантов алгоритма BFGS и L-BFGS на примере функции (8).

2. Практическое применение: использование метода оптимизации BFGS, L-BFGS для обучения модели МО.

   - Адаптировать подход машинного обучения, используемый в Лаб. работе 1, под более вычислительно-дорогое приближение градиента на основе конечно-разностных методов;
   - Реализовать алгоритм оптимизации Adam, адаптированный под конечно-разностные методы определения производных;
   - На основе уже созданных методов оптимизации реализовать обучение модели;
   - Провести сравнение характеристики работы алгоритмов BFGS, L-BFGS и Adam (с различными гиперпараметрами) при решении задачи МО.

Список литературы

[1] Shi, Hao-Jun Michael and Xuan, Melody Qiming and Oztoprak, Figen and Nocedal, Jorge, "On the Numerical Performance of Derivative-Free Optimization Methods Based on Finite-Difference Approximations", arXiv:2102.09762, 2021.

[2] Jorge Nocedal, Stephen J. Wright, "Numerical Optimization", Springer New York, NY, 2006.

[3] Yu. Nesterov, Introductory Lectures on Convex Optimization, Kluwer, Boston, 2004.

[4] Pytlak, Radosław. Conjugate Gradient Algorithms in Nonconvex Optimization, 2009.