Национальный исследовательский университет ИТМО
Факультет информационных технологий и программирования
Прикладная математика и информатика

Методы оптимизации

Отчет по лабораторной работе №3

⟨Собрано 5 июня 2023 г.⟩

Работу выполнили:
Бактурин Савелий Филиппович M32331
Вереня Андрей Тарасович M32331
Сотников Максим Владимирович M32331

Преподаватель:
Ким Станислав Евгеньевич

Решение задачи нелинейной регрессии

Часто решая задачу создания регрессионной модели мы сталкиваемся с тем, что по жизни очень немногие рассматриваемые функции оказываются не представимы в виде обобщенной линейной зависимости или полиномиальной некоторой конечной степени k. Такая же ситуация часто случается и с некоторым набором данных, который нужно как-то обобщить. Именно в таких случаях к нам на помощь приходит более частный случай регрессионного анализа – *нелинейная регрессия.*

Идея построения нелинейной регрессии как и в случае с полиномиальной заключается в том, чтобы найти математическую функцию, которая максимально точно описывает зависимость между независимой переменной и зависимой от нее. Например, для построения нелинейной регрессии можно использовать функции типа полинома, логарифмической или экспоненциальной зависимости.

В целом весь процесс нахождения нелинейной регрессионной модели можно поделить на два этапа:

- ▷ Определить регрессионную модель $f(w, x)$, которая зависит от параметров $w = (w_1,\ \ldots,\ w_W)$ и свободной переменной x.
- ▷ Решить задачу по нахождению минимума сумма квадратов регрессионных остатков:

$$S = \sum_{i=1}^{m} r_i^2,\ r_i = y_i - f(w, x_i)$$

Однако, решая в лоб такую задачу, мы сталкиваемся с оптимизационной задачи нахождения параметров нелинейной регрессионной модели. Тут к нам и приходят на помощь различные методы нахождения, в том числе и рассматриваемые ниже: *Gauss-Newton* и *Powell Dog Leg.*

Gauss-Newton

Напомним, что мы решаем следующую задачу: дана нелинейная модель $f(w, x)$, где $w \in \mathbb{R}^n$, тогда сумма квадратов регрессионных остатков высчитывается как

$$S = \sum_{i=1}^{\texttt{sizeof } X} (f(w, x_i) - y_i)^2 \to \min$$

Итак, теперь введем некоторые новые объекты для решения задачи, пусть $w^0 = (w_0^0,\ w_1^0,\ \ldots,\ w_p^t)$ – начальное приближение, и

$$\begin{aligned}
&\beth = \left(\frac{\partial f}{\partial w_j}(w^t, x_i)\right)_{(\texttt{sizeof } X)\times p} && - \text{ Якобиан, или матрица первых производных}\\
&\vec{f_t} = \big(f(w^t, x_i)\big)_{(\texttt{sizeof } X)\times 1} && - \text{ вектор значений } f\\
&\eth_t = \texttt{const} && - \text{ размера шага}
\end{aligned}$$

Тогда, формула t-й итерации рассматриваемого метода будет высчитываться как

$$w^{t+1} \leftarrow w^t - \eth_t \cdot \underbrace{\left(\beth_t^{\mathrm{T}}\beth_t\right)^{-1}\beth_t^{\mathrm{T}}}_{\Psi}(\vec{f_t} - y),$$

где Ψ – это псевдообратная матрица, или решение некоторой задачи многомерной линейной регрессии, где мы ищем такой вектор Ψ, что

$$\left\| \beth_t \Psi - (\vec{f}_t - y) \right\|^2 \to \min,$$

где y – вектор правильных/настоящих ответов нашей модели. Получается, для решения задачи, мы, так называемую, *невязку* пытаемся приблизить линейной комбинацией вектора из матрицы Якобиана так, что при следующем шаге итерации получить такой w^{t+1}, который бы сократил нам расстояние невязки. Причем, заметим, что на каждом шаге, задача будет новой, так как $\beth_t$ зависит от текущего приближения, чтобы решить задачу многомерной регрессии.

Исследования

Powell Dog Leg

Исследования

BFGS

BFGS, или Алгоритм Бройдена - Флетчера - Гольдфарба - Шанно – это тоже оптимизационный итерационный алгоритм для нахождения локального экстремума для не представимых данных или функций в линейном/полиномиальном виде.

Один из известных квазиньютоновских методов (то есть, тех, которые основаны на получении информации о кривизне функции). Тут следует сразу пояснить, что в квазиньютоновских методах для нахождения оптимальных параметров используется довольно медлительное определение *гессиана* функции (или: матрица вторых производных). И вот тут данный алгоритм ускоряет работу на порядок: ибо он не явно каждый раз высчитывает матрицу, а лишь приближает к ней значения.

Рассмотрим идею этого алгоритма. Пусть дана нам некоторая функция $f(\vec{x})$ и, как обычно, решаем задачу оптимизации нахождения $\underset{\vec{x}}{\operatorname{argmin}} f(\vec{x})$. Пусть также $x_i = \{x_i^0,\ x_i^1,\ \ldots,\ x_i^{n-1}\}$, где n – размерность рассматриваемого пространства; $x_0 \leftarrow \mathbf{INIT}$ — начальная точка; $H_0 = B_0^{-1}$ — начальное приближение, где B_0^{-1} – обратный гессиан функции. Тогда:

0) Пусть i – текущий номер итерации алгоритма.

1) Находим точку, в направлении которой будем производить поиск, она определяется следующим образом:

$$p_i = -H_i \times \nabla f_i$$

2) Вычисляем x_{i+1} через рекуррентное соотношение следующего вида:

$$x_{i+1} = x_i + \alpha \cdot p_i,$$

где α – коэффициент, удовлетворяющий условиям Вольфа, которые, напомню, выглядят вот так:

$$f(x_i + \alpha \cdot p_i) \leqslant f(x_i) + c_1 \cdot \alpha \cdot \nabla f_k^T p_i$$
$$\nabla f(x_i + \alpha \cdot p_i)^T p_i \geqslant c_2 \cdot \nabla f_i^T p_i$$

3) Теперь определим размер шага алгоритма после данной итерации и изменение градиента следующими соответствующими образами:

$$s_i = x_{i+1} - x_i$$
$$y_i = \nabla f_{i+1} - \nabla f_i$$

4) Наконец, обновим гессиан функции, зная, что $\mathbf{I}$ – единичная матрица и $\lambda = \dfrac{1}{y_i^{\mathrm{T}} s_i}$:

$$H_{i+1} = \left(\mathbf{I} - \lambda s_i y_i^{\mathrm{T}}\right) H_i \left(\mathbf{I} - \lambda y_i s_i^{\mathrm{T}}\right) + \lambda s_i s_i^{\mathrm{T}}$$

Исследования

L-BFGS

Исследования