Метод Hessian Free

Оглавление

Теоретическая часть

Введение

Обучение нейронных сетей заключается в минимизации ошибок по отношению к наборам параметров. Как правило, большие нейронные сети могут иметь миллионы параметров, поэтому их обучение является достаточно сложной задачей.

Удивительно, но многие из последних достижений в области нейронных сетей связаны не с повышением скорости обработки данных, улучшением архитектур нейронных сетей, алгоритмов машинного обучения и прочих хитростей, а использованием мощных методов минимизации многомерной функции.

В данной работе будут рассмотрены несколько простых методов локальной минимизации многомерной функции, что позволит, в итоге, подойти к рассмотрению достаточно сложного, но эффективного метода Hessian Free.

Градиентный спуск

Простейшим итерационным алгоритмом локальной минимизации дифференцируемой функции является метод **градиентного спуска**. Градиентом функции $f(x)$ называется следующий вектор из частных производных:

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n}\right)$$

Градиент функции $f(x)$, вычисленный в определенной точке, указывает в направлении максимального роста функции в этой точке, или, что эквивалентно, в противоположном направлении к максимальному уменьшению функции $f(x)$. Формально, алгоритм запишется следующим образом:

Шаг 1. Задается начальное приближение $x^{(0)}$ и требуемая точность ε:

$$x^{(0)} = \left(x_1^{(0)}, x_2^{(0)}, \dots, x_n^{(0)}\right)$$

Шаг 2. Вычисляется градиент функции в текущей точке $x^{(i)}$:

$$v^{(i)} = \nabla f\left(x^{(i)}\right).$$

Шаг 3. Осуществляется перемещение из текущей точки $x^{(i)}$ с некоторым фиксированным шагом α в направлении, противоположном вектору $v^{(i)}$:

$$x^{(i+1)} = x^{(i)} - \alpha v^{(i)}$$

Шаг 4. Проверяется условие достижения необходимой точности ε:

$$\left|f\left(x^{(i+1)}\right) - f\left(x^{(i)}\right)\right| < \varepsilon$$

Если условие не выполняется, то осуществляется переход к следующей итерации на шаг 2, иначе $x^{(i+1)}$ – найденная точка минимума с точностью ε.

Существенный недостаток этого метода заключается в том, что метод градиентного спуска является методом первого порядка, то есть, в нем учитываются только первые частные производные функции $f(x)$, тогда как функция $f(x)$ может иметь достаточно сложный нелинейный вид.

Ограничение в виде игнорирования поверхностей ошибок, по своей структуре более сложных, чем плоскостей, не позволяет применять данный метод для быстрого и эффективного обучения нейронных сетей.

Метод Ньютона

Данный метод учитывает вторые производные функции $f(x)$, поэтому этот метод является методом второго порядка. Рассмотрим минимизацию методом Ньютона применительно к одномерной функции, а потом обобщим на многомерный случай.

Одномерный случай

Предположим, что функция $f(x)$ имеет минимум. Разложим функцию $f(x)$ в ряд Тейлора в точке x_0:

$$f(x_0 + x) \approx f(x_0) + f'(x_0)x + \frac{f''(x_0)}{2}x^2 + O(x^3)$$

Мы хотим найти точку x, в которой первая производная функции в точке x равна нулю (условие экстремума функции. Используя для этого разложение функции в ряд Тейлора и отбросив члены третьего порядка и больше, получим:

$$\frac{d}{dx}\left(f(x_0) + f'(x_0)x + \frac{f''(x_0)}{2}x^2\right) = f'(x_0) + f''(x_0)x = 0 \Rightarrow x = -\frac{f'(x_0)}{f''(x_0)}$$

Если $f(x)$ просто квадратичная функция, то это будет абсолютный минимум. Для нахождения минимума любой нелинейной функции необходимо использовать итерационный процесс, вычисляя на каждой итерации следующее приближение $x^{(i+1)}$ по формуле:

$$x^{(i+1)} = x^{(i)} - \frac{f'\left(x^{(i)}\right)}{f''(x^{(i)})} = x^{(i)} - \left(f''\left(x^{(i)}\right)\right)^{-1}\left(f'\left(x^{(i)}\right)\right)$$

Если минимум существует, то алгоритм итерационно приблизится к нему с требуемой точностью.

Многомерный случай

Предположим, что для функции $f(x)$ существует минимум. В многомерном случае применяется аналогичный подход: производная эквивалентна градиенте, а вторая производная эквивалентна матрице Гессе – матрице вторых производных:

$$f'(x) \to \nabla f(x)$$

$$f''(x) \to H(f)(x)$$

где:

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}\right)$$

$$\mathrm{H}(f) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{pmatrix}$$

Тогда формула для получения следующего приближения запишется как:

$$\mathrm{x}^{(i+1)} = x^{(i)} - \left(H(f)\left(x^{(i)}\right)\right)^{-1} \nabla f\left(x^{(i)}\right)$$

Проблемы метода Ньютона

Метод Ньютона является методом второго порядка, поэтому может работать значительно лучше, чем градиентный спуск. Предположение, что функция $f(x)$ является квадратичной, позволяет осуществлять большие шаги по направлению к минимуму при низкой кривизне (когда значение $f''\left(x^{(n)}\right)$ мало), и маленькие шаги при большой кривизне. Как и в методе градиентного спуска, перемещение от текущего приближения к следующему осуществляется по направлению, противоположному направлению максимального роста функции ($-\nabla f\left(x^{(n)}\right)$).

Однако, метод Ньютона имеет очень большой недостаток: он требует вычисления матрицы Гессе, что, в свою очередь, требует $O(n^2)$ по памяти и времени, и вычисления обратной к ней, что можно реализовать методом Гаусса с прямым и обратным ходом за $O(n^3)$ по времени.

В дальнейшей работе будет показано, что метод Hessian Free лишен этих недостатков, хотя и использует основную идею метода Ньютона – приближение функции $f(x)$ ее рядом Тейлора до членов второго порядка включительно.

Метод сопряженных градиентов

Пусть $f(x)$ – квадратичная функция вида:

$$\mathrm{f}(x) = \frac{1}{2} x^T A x + b^T x + c, \text{где } A \in \mathbb{R}^{n \times n}, x, b, c \in \mathbb{R}^n$$

Любую квадратичную форму мы можем преобразовать к такому виду, что $A^T = A$: выпишем квадратическую форму в виде суммирования, посчитаем коэффициенты при $x_i x_j$, поделим коэффициенты на два и составим из них симметрическую матрицу A.

Градиент находится как:

$$\nabla f(x) = Ax + b$$

Пусть $x^{(0)}$ – начальное приближение. Направление, в котором находится следующее приближение, противоположно градиенту:

$$d^{(0)} = -\nabla f\left(x^{(0)}\right)$$

Тогда следующее приближение вычисляется по формуле, аналогичной в методе градиентного спуска:

$$\mathrm{x}^{(1)} = x^{(0)} + \alpha d^{(0)}$$

В отличие от обычного градиентного спуска, где шаг был фиксированным числом, в наискорейшем градиентном спуске на каждой итерации вычисляется оптимальный шаг. Вводится следующая функция, зависящая от α:

$$g(\alpha) = f\left(x^{(i)} + \alpha d^{(i)}\right) = \frac{1}{2}\left(x^{(i)} + \alpha d^{(i)}\right)^T A\left(x^{(i)} + \alpha d^{(i)}\right) + b^T\left(x^{(i)} + \alpha d^{(i)}\right) + c =$$

$$= \frac{1}{2}\alpha^2\left(d^{(i)}\right)^T A d^{(i)} + \left(d^{(i)}\right)^T\left(Ax^{(i)} + b\right)\alpha + \left(\frac{1}{2}\left(x^{(i)}\right)^T Ax^{(i)} + \left(x^{(i)}\right)^T d^{(i)} + c\right)$$

Вычисление оптимального шага эквивалентно минимизации функции $g(\alpha)$. Предположим, что она имеет минимум, тогда этот минимум является глобальным (т.к. функция квадратическая). Условие на минимум:

$$g'(\alpha) = 0 = \left(\left(d^{(i)}\right)^T A d^{(i)}\right)\alpha + \left(d^{(i)}\right)^T\left(Ax^{(i)} + b\right) \Rightarrow \alpha = -\frac{d^{(i)}\left(Ax^{(i)} + b\right)}{(d^{(i)})^T A d^{(i)}}$$

Далее необходимо учесть тот факт, чтобы следующее направление было сопряжено с текущим направлением, иначе могут возникать ситуации, в которых направления будут противоположны, поэтому мы требуем условие сопряженности.

Два вектора x и y сопряжены, если верно следующее условие: $x^T A y = 0$.

Следующее направление $d^{(1)}$, находится по формуле:

$$d^{(1)} = -\nabla f\left(x^{(1)}\right) + \beta_0 d^{(0)}$$

Здесь β_0 выбирается из условия сопряженности на вектора $d^{(1)}$ и $d^{(0)}$:

$$d^{(1)} A d^{(0)} = 0 \Rightarrow \left(d^{(1)}\right)^T A d^{(0)} = -\left(\nabla f\left(x^{(1)}\right)\right)^T A d^{(0)} + \beta_0\left(d^{(0)}\right)^T A d^{(0)} = 0 \Rightarrow \beta_0 = \frac{\left(\nabla f\left(x^{(1)}\right)\right)^T A d^{(0)}}{(d^{(0)})^T A d^{(0)}}$$

Стоит заметить, что вычисление оптимального β_i позволяет получить направление, сопряженное со всеми предыдущими направлениями, поэтому достаточно сделать n итераций, где n – размерность пространства.

Тогда метод сопряженных градиентов для квадратичных функций формально можно записать следующим образом:

Пусть $f(x)$ – квадратичная функция вида $f(x) = \frac{1}{2}x^T Ax + b^T x + c$.

Шаг 1: Инициализация. Положим $i = 0$, $x^{(i)} = x^{(0)}$ и вычислим $\mathrm{d}^{(i)} = d^{(0)} = -\nabla f\left(x^{(0)}\right)$.

Шаг 2: Поиск оптимального шага. Вычислим α как результат минимизации функции $f\left(x^{(i)} + \alpha d^{(i)}\right)$ по формуле:

$$\alpha = -\frac{\left(d^i\right)^T\left(Ax^{(i)} + b\right)}{(d^{(i)})^T A d^{(i)}}$$

Шаг 3. Обновление приближения. Вычислим $x^{(i+1)} = x^{(i)} + \alpha d^{(i)}$

Шаг 4. Обновление направления. Вычислим $d^{(i+1)} = -\nabla f\left(x^{(i+1)}\right) + \beta_i d^{(i)}$, где β_i находится как:

$$\beta_i = \frac{\left(\nabla f\left(x^{(i+1)}\right)\right)^T A d^{(i)}}{(d^{(i)})^T A d^{(i)}}$$

Шаг 5. Итерационный процесс. Повторим шаги 2-4 пока мы не рассмотрим n направлений d_i, где n – размерность пространства.

Метод Hessian Free

Алгоритмически метод можно записать в следующей форме:

Пусть задана функция $f(x)$: $\mathbb{R}^n \to \mathbb{R}$, которую необходимо минимизировать, ε – необходимая точность, и точка $x^{(0)}$ – начальное приближение.

Шаг 1. Инициализация. Положим $i = 0$ и $x^{(i)} = x^{(0)}$.

Шаг 2. Аппроксимация квадратичной функцией. Для текущего приближения $x^{(i)}$ вычислим градиент $\nabla f\left(x^{(i)}\right)$ и матрицу Гессе $H(f)\left(x^{(i)}\right)$, и будем иметь в виду, что для функции $f(x)$ справедливо разложение в ряд Тейлора:

$$f(x + \Delta x) \approx f(x) + \left(\nabla f(x)\right)^T \Delta x + (\Delta x)^T H(f) \Delta x + O(\|\Delta x\|^3)$$

Шаг 3. Сопряженный градиент. Вычислим $x^{(i+1)}$ используя метод сопряженных градиентов для квадратичной функции (для текущего разложения в ряд Тейлора). В качестве переменной для метода сопряженных градиентов служит Δx.

Шаг 4. Итерационный процесс. Повторим шаги 2 и 3 пока $x^{(i)}$ пока процесс не сойдется с необходимой точностью ε.

Данный алгоритм вбирает в себя все описанные ранее идеи. Рассмотрим обход вычисления матрицы Гесса на шаге 2.

Обход вычисления матрицы Гессе в методе Hessian Free

В методе Ньютона была необходимость в вычислении матрицы Гессе, но в данном алгоритме вычисление матрицы Гессе не требуется, так как во всех формулах используется результат умножения матрицы Гессе на вектор.

Рассмотрим $i-$ ый элемент вектора, получающегося после произведения матрицы Гессе на вектор:

$$(H(f)v)_i = \sum_{j=1}^{N} \frac{\partial^2 f}{\partial x_i \partial x_j}(x) \cdot v_j = \nabla \frac{\partial f}{\partial x_i}(x) \cdot v$$

Следует заметить, что это ни что иное как производная по направлению от $\frac{\partial f}{\partial x_i}$ в направлении v:

$$\nabla_v f = \lim_{\varepsilon \to 0} \frac{f(x + \varepsilon v) - f(x)}{\varepsilon}$$

Используя разностную аппроксимацию дифференцирования для малого шага h, получим приближение с точностью $O(\|hv\|)$ или $O(\|hv\|^2)$:

$$\nabla_v f \approx \frac{f(x+hv) - f(x)}{h} + O(|h|\|v\|^2)$$

$$\nabla_v f \approx \frac{f(x+hv) - f(x-hv)}{2h} + O(|h|^2\|v\|^3)$$

При желании, можно доказать эти формулы, используя разложение Тейлора функции f(x).

Тогда аппроксимацию умножения матрицы Гессе на вектор можно вычислить как:

$$Hv \approx \frac{\nabla f(x+hv) - \nabla f(x)}{h} + O(|h|\|v\|^2)$$

$$Hv \approx \frac{\nabla f(x+hv) - \nabla f(x-hv)}{2h} + O(|h|^2\|v\|^3)$$