# Метод Hessian Free

Оглавление

[Теоретическая часть 4](#_Toc501618488)

[Введение 4](#_Toc501618489)

[Градиентный спуск 4](#_Toc501618490)

[Метод Ньютона 5](#_Toc501618491)

[Одномерный случай 5](#_Toc501618492)

[Многомерный случай 5](#_Toc501618493)

[Проблемы метода Ньютона 6](#_Toc501618494)

[Метод сопряженных градиентов 6](#_Toc501618495)

[Метод Hessian Free 8](#_Toc501618496)

[Обход вычисления матрицы Гессе в методе Hessian Free 8](#_Toc501618497)

# 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 называется следующий вектор из частных производных:

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

**Шаг 1.** Задается начальное приближение и требуемая точность :

**Шаг 2.** Вычисляется градиент функции в текущей точке

**Шаг 3.** Осуществляется перемещение из текущей точки с некоторым фиксированным шагом в направлении, противоположном вектору :

**Шаг 4.** Проверяется условие достижения необходимой точности :

Если условие не выполняется, то осуществляется переход к следующей итерации на шаг 2, иначе – найденная точка минимума с точностью .

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 , тогда как функция 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 , 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 имеет минимум. Разложим функцию в ряд Тейлора в точке :

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

Если просто квадратичная функция, то это будет абсолютный минимум. Для нахождения минимума любой нелинейной функции необходимо использовать итерационный процесс, вычисляя на каждой итерации следующее приближение по формуле:

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

### Многомерный случай

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

где:

Тогда формула для получения следующего приближения запишется как:

## Проблемы метода Ньютона

Метод Ньютона является методом второго порядка, поэтому может работать значительно лучше, чем градиентный спуск. Предположение, что функция является квадратичной, позволяет осуществлять большие шаги по направлению к минимуму при низкой кривизне (когда значение 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 ().

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

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

## Метод сопряженных градиентов

Пусть – квадратичная функция вида:

Любую квадратичную форму мы можем преобразовать к такому виду, что выпишем квадратическую форму в виде суммирования, посчитаем коэффициенты при поделим коэффициенты на два и составим из них симметрическую матрицу

Градиент находится как:

Пусть – начальное приближение. Направление, в котором находится следующее приближение, противоположно градиенту:

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

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

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

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 и сопряжены, если верно следующее условие:

Следующее направление , находится по формуле:

Здесь выбирается из условия сопряженности на вектора и

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

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

Пусть – квадратичная функция вида .

**Шаг 1: Инициализация.** Положим , и вычислим

**Шаг 2: Поиск оптимального шага.** Вычислим как результат минимизации функции по формуле:

**Шаг 3. Обновление приближения.** Вычислим

**Шаг 4. Обновление направления.** Вычислим , где находится как:

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

## Метод Hessian Free

Алгоритмически метод можно записать в следующей форме:

Пусть задана функция , которую необходимо минимизировать, – необходимая точность, и точка – начальное приближение.

**Шаг 1. Инициализация.** Положим и .

**Шаг 2. Аппроксимация квадратичной функцией.** Для текущего приближения вычислим градиент и матрицу Гессе , и будем иметь в виду, что для функции справедливо разложение в ряд Тейлора:

**Шаг 3. Сопряженный градиент.** Вычислим используя метод сопряженных градиентов для квадратичной функции (для текущего разложения в ряд Тейлора). В качестве переменной для метода сопряженных градиентов служит .

**Шаг 4. Итерационный процесс.** Повторим шаги 2 и 3 пока пока процесс не сойдется с необходимой точностью .

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 элемент вектора, получающегося после произведения матрицы Гессе на вектор:

Следует заметить, что это ни что иное как производная по направлению от в направлении :

Используя разностную аппроксимацию дифференцирования для малого шага , получим приближение с точностью или :

При желании, можно доказать эти формулы, используя разложение Тейлора функции .

Тогда аппроксимацию умножения матрицы Гессе на вектор можно вычислить как: